REAL TIME IMPLEMENTATION OF AN LPC ALGORITHM. SPEECH SIGNAL PROCESSING RESEARCH AT CHI

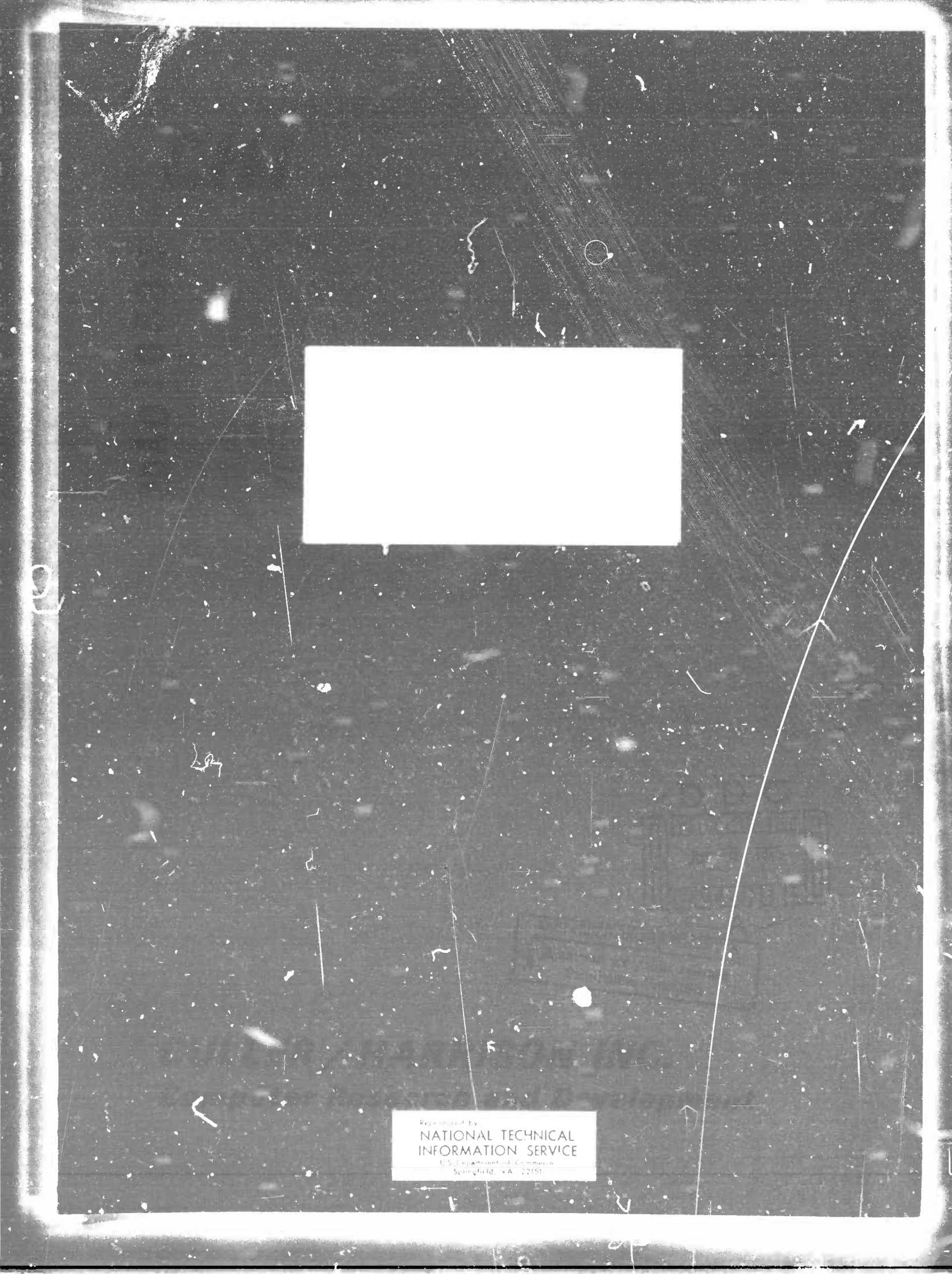
Glen J. Culler, et al

Culler/Harrison, Incorporated

Prepared for:

Defense Advanced Research Projects Agency

May 1975



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CHI-QTR-101	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER AD-A010 532
4. TITLE (and Subtitle) REAL TIME IMPLEMENTATION OF AN LPC ALGORITHM Quarterly Technical Report Speech Signal Processing Research at CHI		5. TYPE OF REPORT & PERIOD COVERED Quarterly Technical Report November 1974 - April 1975 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Glen J. Culler, Michael McCammon, James F. McGill		8. CONTRACT OR GRANT NUMBER(s) DAHC15 73 C 0252
9. PERFORMING ORGANIZATION NAME AND ADDRESS Culler/Harrison, Inc. 150-A Aero Camino Goleta, California 93017		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program Code: P5P10
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Supply Service -- Washington Room 1D 245, The Pentagon Washing, D.C. 20310		12. REPORT DATE May 1975 13. NUMBER OF PAGES 97
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Defense Contract Administration Services District, Van Nuys 18321 Ventura Blvd. Tarzana, California 91356		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 2359.

PRICES SUBJECT TO CHANGE

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Speech Compression
Speech Analysis - Synthesis
Linear Predictive Coding
Vocoders
Acoustic Data Collection
ARPANET

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report covers the work done for ARPA by CHI on Contract No. DAHC15 73 C 0252 during the period November 1974 - April 1975. During the contract period reported here, we have implemented a real time Linear Predictive coding algorithm for speech data compression developed by Markel and Gray at SCRL. Chapter 1 of this report documents this implementation and reports on its use over the ARPANET. Data collection equipment for a physical acoustics laboratory was designed and prototyped. Both the prototype and the multiplexed data collection system are documented in Chapter 2 of this report.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

CULLER/HARRISON, INC.
150-A Aero Camino
Goleta, California 93017
(805) 968-1064

May 1975

REAL TIME IMPLEMENTATION of an LPC ALGORITHM

Quarterly Technical Report

Speech Signal Processing Research at CHI

November 1974 - April 1975

PRINCIPAL INVESTIGATOR: Dr. Glen J. Culler

PROJECT SCIENTISTS: Dr. Michael McCammon
Dr. James F. McGill

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 2359 Contract No. DAHC15 73 C 0252

CONTENTS

LIST OF FIGURES

LIST OF TABLES

SUMMARY

This report covers the work done for ARPA by CHI on Contract No. DAHC15 73 C 0252 during the period November 1974 - April 1975.

The development of digital communication systems which include spoken messages requires that we be able to extract from the speaker's voice a numerical representation that can be transmitted through the system and reconstructed with enough fidelity that the listener can properly understand the message.

There are many technical issues involved here and ARPA's program of research in Network Speech Compression coordinates the work of several contractors in a cooperative attack on these issues.

The contract work at CHI engages two aspects of this overall technical problem.

A. To provide real time implementation of selected algorithms in an operational form on the ARPANET.

B. To perform experiments in physical accustics designed to provide a basis for improving these algorithms.

During the contract period reported here, we have implemented an algorithm developed by Markel and Gray at SCRL. Chapter 1 of this report documents this implementation and reports on its use over the ARPANET. It was demonstrated over the ARPANET at the December meeting of the Network Speech Compression Group. Delays imposed by network's handling of the packetized messages were of the order of three seconds. While these delays did not destroy the communication, they were too inconvenient for the user to be satisfied with. More recent experiments using uncontrolled network messages that minimize network overhead have brought these delays down to a more reasonable one second.

One of the important consequences of using a real time implementation is that algorithmic shortcomings are able to be judged in an appropriate context. Simulations using test data sets are valuable for algorithmic development but the rich variety of signals produced in an operational environment are the ultimate test of the algorithm. Our experience with this first algorithm is summarized by:

a. The reconstructed signal generally does an excellent job of sounding like the speaker and conveys the intended message.

b. Whenever the quality of the input signal is lowered by spurious noises the output does not degrade as gracefully as we would like. There is a tendency to pop and crackle.

c. The pitch extraction part of the algorithm provides good results for the most part, but degrades too easy for low amplitude inputs.

d. For high-pitched female voices, there is an artificial increase in gain for some sounds.

Some improvements have been made in relationship to these observations, but more needs to be done to bring the algorithm to its best form.

The work performed for task B. has been of a preparatory nature. The data collection equipment for the laboratory was designed and prototyped. This prototype was used to improve the analog aspects of the real time system. Both the prototype and the multiplexed data collection system are documented in Chapter 2 of this report.

Our acoustical experiments and their implications for numerical representation of speech will be the major substance of our next quarterly report.

CHAPTER 1. A REAL-TIME SPEECH COMPRESSION SYSTEM ON THE ARPA NETWORK

As part of our responsibilities within the Network Speech Compression group we have implemented a system for real time communication over the ARPA network using linear predictive coding for speech data compression. This system is implemented on the Culler/Harrison MP-32A signal processor and AP-90 array/arithmetic processor.

The first section of this chapter will discuss the real time linear prediction vocoder. The following sections will describe the total computer system which supports the use of this vocoder over a packet network and examine in some detail our experience with uncontrolled network messages. We conclude this chapter with some examples of the speech quality obtained.

1.1 THE LINEAR PREDICTION VOCODER

The speech compression system which we have developed uses the autocorrelation method of linear prediction. The approach we have followed was developed by John Markel of the Speech Communication Research Laboratory and was adopted for use by the Network Speech Compression group [4,8].

Analysis and synthesis are performed independently as part of the transmitter and receiver, respectively, of a speech compression system. Figures 1-1 and 1-2 show block diagrams of our transmitter and receiver systems.

Linear predictive speech analysis is composed of two parts. The first is coefficient analysis to obtain a set of inverse filter coefficients that describe the spectral behavior of the speech waveform and a residual energy or gain for each frame of data analyzed. The second is a pitch detector which determines the period of the driving function or the absence of voicing for each frame. The parameters resulting from this analysis are, therefore, pitch (or unvoiced), gain, and filter coefficients. These parameters are quantized for efficient transmission at low bit rates.

The receiver decodes the quantized parameters. It then computes a synthetic speech signal using a pitch excited filter.

In order to clarify the discussion of our analysis and synthesis implementations, the next part of this section describes the hardware capabilities of the MP-32A and AP-90 processors.

The analysis and synthesis routines are discussed in detail in the remainder of this section. We have also included a description of the modifications and refinements which we have made to the algorithms and programs described by Markel and Gray [2,3,4,5].

Throughout this section illustrations and timings will refer to the 150 microsecond sampling rates system used for the initial network experiments. Timing formulae and instruction count information for general cases are given in Table 1-2 at the end of the section.

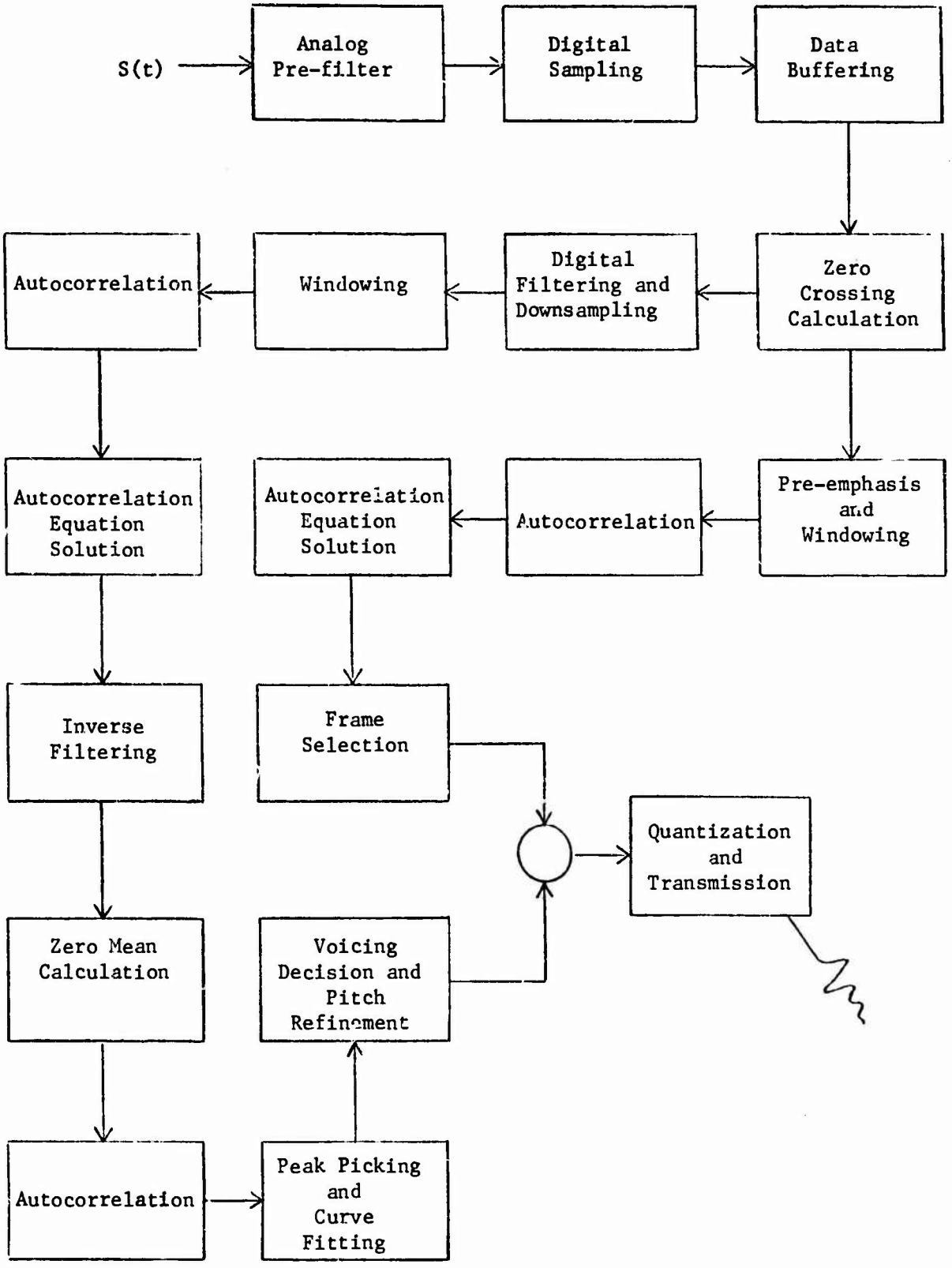


Figure 1-1: Transmi ter Block Diagram

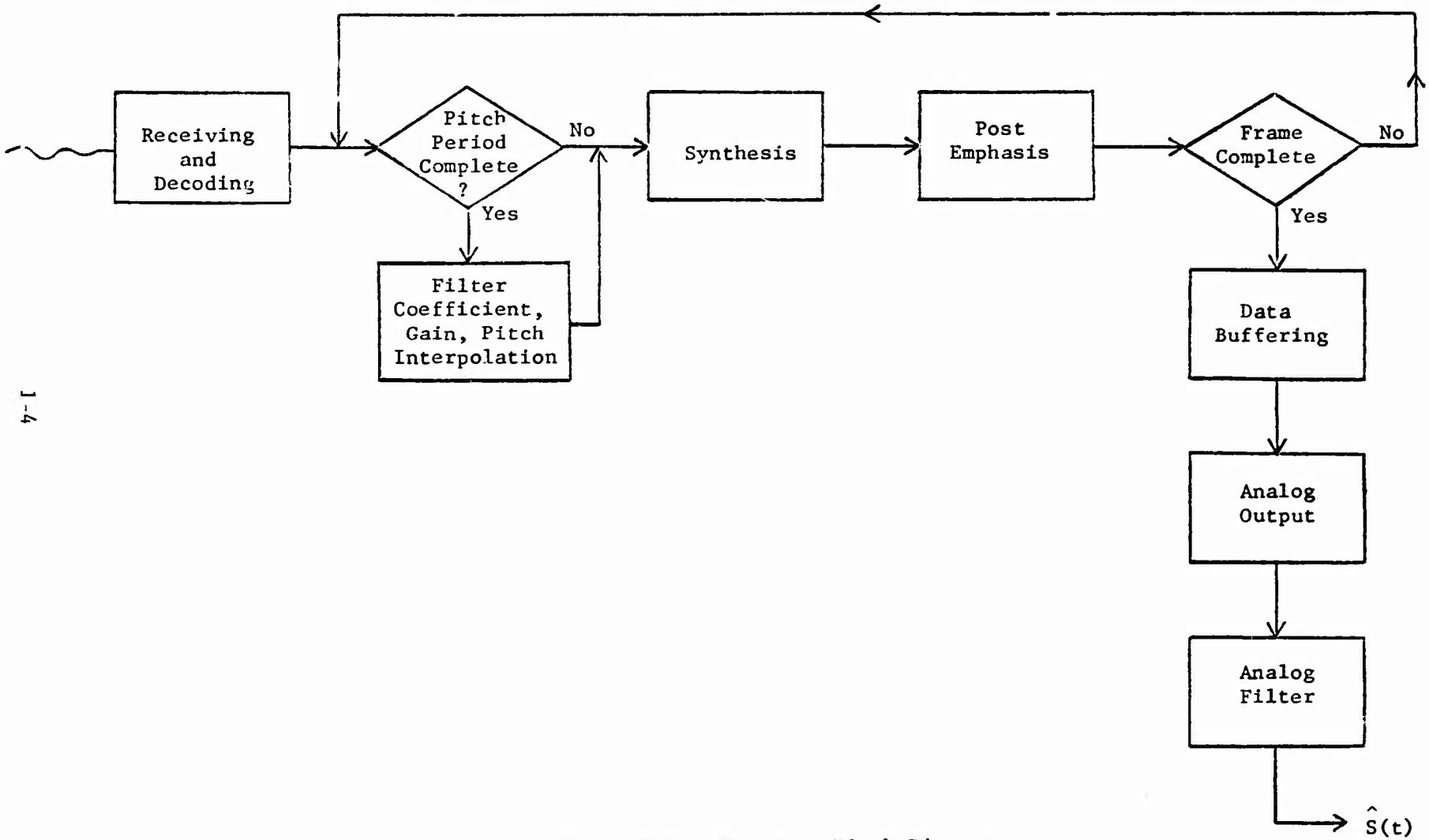


Figure 1-2. Receiver Block Diagram

1.1.1 HARDWARE DESCRIPTION

The MP-32A computer is a micro-programmable processor with 16-bit addition, subtraction and logical operations and an 8 x 8 multiplier. It has a large, fast MOS memory (64K x 18 bits, 500 nanosecond cycle time interleaved four ways). All input/output devices except the ARPA net Very Distant Host interface and the current analog I/O system are interfaced to the MP-32A. For the network voice system, it has responsibility for scheduling the array processor and itself, buffering all data except that currently being processed in one of the other processors, and all aspects of the network voice protocol. These roles will be discussed in the next section. Its primary responsibility in terms of the linear predictive analysis and synthesis is providing data and program overlays for the array processor, performing the pitch and voicing decisions based on the peak position and correlation calculated by the array processor and encoding and decoding of the analysis parameters.

The AP-90 array processor is an arithmetic processor with its own MOS memory for data and separate program memory (PS), data pad (DP), index pad (SP) and fixed data memory (ROM). Figure 1-3 illustrates the components of the AP. The data pad is 32 32-bit registers which normally forms one input to arithmetic operations. The other input and the destination for arithmetic operations is the 32-bit accumulator. Index pad provides 16 12-bit registers used for counting and indexing MOS memory or data pad. The fixed data memory provides tables of elementary function values and a folded 4096th order roots of unity table. The program store holds 512 32-bit instructions.

Unlike some other array processors, the AP-90 uses a single instruction stream. Each instruction, however, may separately control data transfer between registers, arithmetic operations and indexing or counting operations. Each instruction takes 167 nanoseconds.

The arithmetic section of the array processor uses three different data lengths. Addition can be done using 16, 24 or 32-bit data. Multiplication can be 16 x 16 or 24 x 24. The 16 x 16 cases can produce either a full 32-bit result or a rounded most significant 16-bit result. The 24 x 24 multiply produces a most significant 24-bit rounded result.

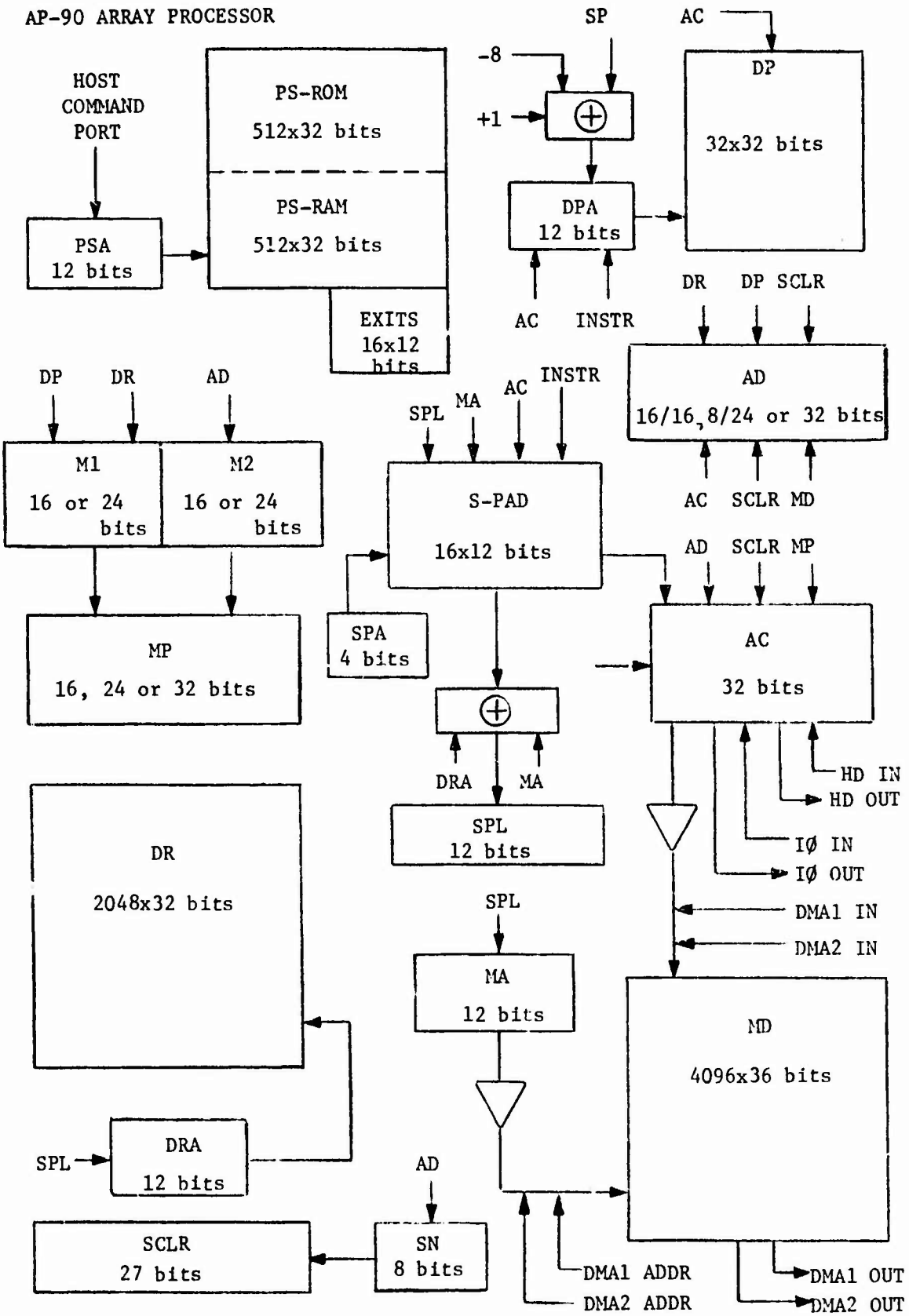


Figure 1-3. AP-90 Processor

Sixteen-bit data may be stored in either the left or right half of each 32-bit register. All forms of the arithmetic are used at present in the AP programs for linear predictive analysis and synthesis.

Transfers of information between the array processor and the MP main memory can take place at rates up to one 32-bit word every 167 nanoseconds. This speed is used to rapidly load program overlays into the AP program store. Data transfer is generally limited to the access time of the AP buffer memory, which is 500 nanoseconds. Control of AP by the MP is provided through a host command path, which allows the MP to initiate the AP at any instruction in program store. The AP can interrupt the MP to provide synchronization.

1.1.2 THE TRANSMITTER SYSTEM

Data Collection: The speech signal is input using a high quality dynamic microphone (SONY ECM280). The analog signal is amplified and filtered by a low pass filter with a cutoff frequency of 3,200 Hz. It is then digitally sampled at a 150 microsecond rate by a 12-bit analog-to-digital converter for input to the digital computer system. The data is collected in blocks of 128 points (19.2 milliseconds time period) for further processing. In our present implementation, this data collection is performed in a separate processor which transfers the data to the MP-32A on a block basis.

Coefficient analysis is performed on 128 point (19.2 millisecond) data frames, pitch and voicing calculations use a 256 point (38.4 ms) frame. Both calculations are performed every 64 points (9.6 msec) to provide finer resolution parameters for input to the quantization and encoding processes. The data management to provide overlapped 128 point data blocks for the coefficient analysis is performed in the MP-32A prior to transfer of the data to the AP-90. The pitch calculation program in the AP-90 uses the second half of each analysis data block, together with points collected from the previous three analysis frames, to form its input data frame.

1.1.2.1 Coefficient Analysis

The autocorrelation method of linear predictive analysis consists of the solution of the set of normal equations:

$$\sum_{j=1}^{M} A_j R_{|i-j|} = -R_i \qquad i=1,\cdots,M$$

where the R_i are the autocorrelation coefficients of the input signal S:

$$R_i = \sum_{j=0}^{N-|i|} s_j s_{j+|i|} \cdot [1]$$

These normal equations can be solved recursively using Levinson's method to give filter coefficients A_j for a "whitening filter". As a byproduct of the solution for the A's, the reflection coefficients or partial autocorrelation coefficients, or K parameters, are also obtained. Since the reflection coefficients have superior quantization properties, they were selected for use by the Speech Compression Group as the parameter for transmission. [1,2,4]

The input data frame for analysis is first passed through a preemphasis filter

$$P=1-\mu z^{-1}$$

with a fixed preemphasis μ which is nominally 0.9, but may be set to any other value between 0 and 1. The filtered data is then Hanning windowed. The purpose of the preemphasis filtering is to reduce the spectral dynamic range over the time frame of the data. The fixed preemphasis factor is a compromise agreed to by the Network Speech Compression group to simplify transmission of parameters as well as the processing itself.

The windowing provides increased resolution in the autocorrelation calculation which is performed in the array processor by program DIFFWIN. The computation carried out is:

$$X_0 = 0$$

$$X_n = 0.5\left(1-\cos(2\pi n/128)\right) \quad (S_n-\mu\, S_{n-1}), \quad n=1,\cdots,127.$$

The input signal data S_n is not normalized prior to this operation, as we have found that the rounded fixed point multiplication hardware in the AP-90 causes no significant loss in quality. The computation time for this calculation is 214 microseconds.

The first eleven autocorrelation coefficients are calculated using double precision multiplication and addition to collect a full 32-bit sum. Since the original input data was limited to 11 bits plus sign, the accumulation of 128 products cannot exceed 30 bits, and therefore, no loss in accuracy is introduced. The resulting sums are scaled individually as 24-bit fractions and all except R_o are normalized by division by R_o. The computation is performed by program AUTOCORR in the array processor using the iteration formulae:

$$R_{i,o} = 0$$

$$R_{i,n+1} = R_{i,n} + X_{n+1} X_{i+n+1} \quad i = 0,10;\ n = 0,127$$

$$R_o = R_{o,128}$$

$$R_i = R_{i,128}/R_o \quad i=1,10$$

The computation time is 868 microseconds. A detailed examination of this program is given in the appendix to this section.

The autocorrelation normal equations to be solved are:

$$\sum_{j=1}^{M} A_j R_{|i-j|} = -R_i, \quad i=1,M.$$

The algorithm used is based on Levinson's method using normalized autocorrelation coefficients as produced by AUTOCORR.

Initialize: $K_1 = -R_1, A_1 = K_1, \alpha = 1-R_1^2$

Loop: DO for n=1, M-1

$$\beta = R_{n+1} + \sum_{j=1}^{n} A_{n-j+1} R_j$$

$$K_{n+1} = -\beta/\alpha$$

$$A_{n+1} = K_{n+1}$$

$$A_{n-j+1} = A_{n-j+1} + K_{n+1} A_j \text{ and } A_j = A_j + K_{n+1} A_{n-j+1}$$

$$\text{for } j = 1, [(n+1)/2]$$

$$\alpha = \alpha + K_{n+1} \beta.$$

The K's calculated as intermediate results through the calculation are the reflection coefficients, which are the parameters used for transmission rather than the A's. The variable α represents the normalized prediction error.

$$E = 1 + \sum_{j=1}^{M} A_j R_j = \prod_{j=1}^{M} (1-K_j^2).$$

When multiplied by the input signal energy R_o and corrected for the loss in energy due to the Hanning window, α is the square of the transmitted gain parameter.

The solution of the autocorrelation equations is performed in the array processor by program SOLVE. This program maintains 24-bit accuracy for all intermediate results through the calculation in order to minimize the occurrence of instabilities due to limited word length. A stability test, consisting of a check that each K parameter is less than one in magnitude, is included. If any K fails this test, all K's from that K on in the computation are forced to 0, as suggested by Markel [6]. The computation time for solving for ten K parameters and αR_o is 235 microseconds.

The final arithmetic processing of the coefficient analysis includes the conversion of each K parameter to a 16-bit number with a fixed scale of 2^{-15}. The conversion from 24-bit normalized form to 16-bit fixed scale uses rounding arithmetic to preserve as much accuracy as possible. The limitation to 16 bits is to simplify encoding, which is performed in the MP-32A, which has only 16-bit arithmetic. The unnormalized energy αR_0 is converted to the desired gain by calculating its square root, although the step is not needed if encoding is to be done, since the encoding tables can use the energy directly. [7] This processing is performed in the AP-90 by program FIXK; it takes 25 microseconds for the 10 K's and gain.

1.1.2.2 Pitch Analysis

The pitch analysis requires an approximately 40 millisecond window, in order to include at least two full pitch periods for all detectable pitches. However, the data is downsampled by a factor of three to approximately a 1KHz Nyquist frequency, so the window length is 86 points. This downsampling must be preceded by digital low pass filtering to avoid aliasing.

In order to avoid extra data transfers, but also maintain proper continuity across block boundaries, the second half of each 128 word data block is digitally filtered and downsampled into a 128 entry circular buffer. The extra length byond the 86 points used is needed since the data is packed two entries per 32-bit computer word. It simplifies the data management needed by exactly holding the down sampled data from six blocks. This is illustrated in Figure 1-4.

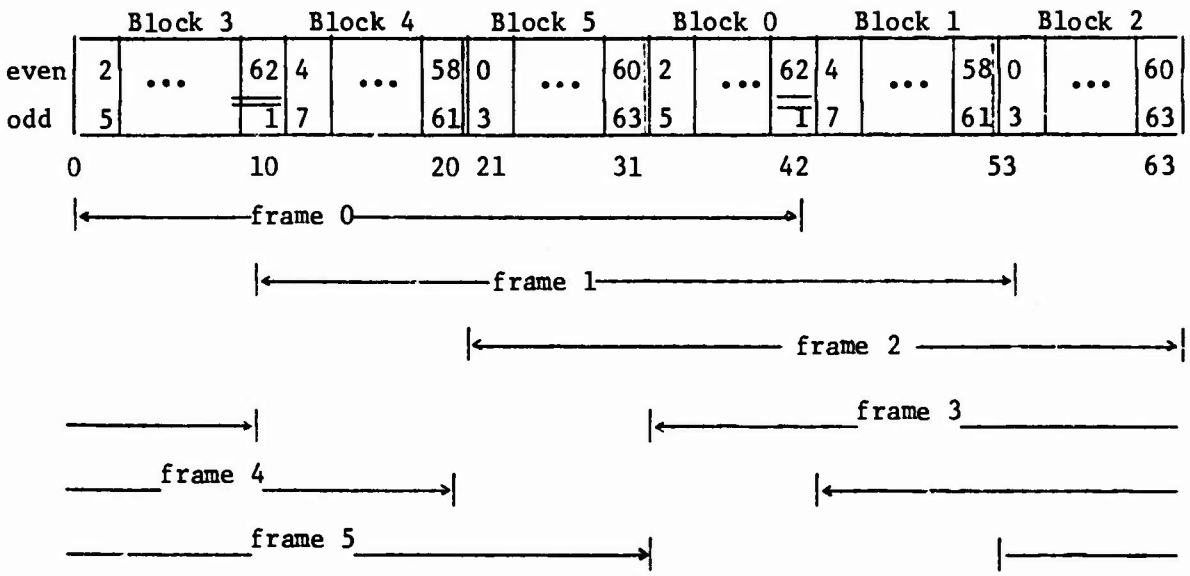


Figure 1-4: Down Sampled Data Buffer

Numbers inside boxes represent the index of sampled points in second half of the 128 word input block.

The digital filter used is a two-multiplier lattice filter of identical structure to the synthesizer filter, but with only five reflection coefficients. The filter algorithm is given in the discussion of the synthesis. The filter coefficients were chosen to give a sharp cutoff at about 1KHz, with reasonably flat response from 100 to 900 Hertz. The filter memory is maintained from block to block to maintain continuity.

In order to properly sequence the downsampling and storing of the data, two global values are maintained. These are the downsample counter and the circular buffer write pointer. No even/odd pointer is necessary. since the words keep the same even/odd parity as long as the downsampling factor is odd.

The digital low pass filtering and downsampling are performed in the AP-90 by the program DOWN, which takes approximately 430 microseconds for each 64 word input block sampled. Sixteen bit rounded arithmetic is used for these calculations. This step of the pitch analysis precedes the preemphasis and windowing step of the coefficient analysis in order to minimize the number of data buffers required in the array processor memory.

The remainder of the pitch analysis is performed after the completion of the coefficient analysis. It could logically have been made conditional by testing either the computed gain or the number of zero crossings as suggested by Markel. [4] We have not implemented any such choice at this time, primarily because the time required for all processing in the AP-90 is only slightly over half that available. If additional processing is needed, as will be the case for 10KHz or higher sampling rates, such tests can be easily incorporated.

The downsampled data is prepared for autocorrelation by applying a Hanning window to the data in the circular buffer and transferring the windowed result to a second linear buffer extended with zeros. The window calculation is:

$$W_j = \left(1-\cos\left(2\pi j\frac{48}{4096}\right)\right) X_j \qquad j=0,\cdots,85$$

The values for the cosine are obtained from a ROM table of the 4096th order roots of unity. An additional global pointer is required to point to the starting location in the circular buffer for the frame to be windowed. The calculation is performed using 16-bit rounded arithmetic in the AP by program WINDOW. The time required is 125 microseconds.

The pitch analysis method used is Markel's simplified inverse filter tracking algorithm (SIFT). [2,5] This method first extracts an error signal from the windowed data by filtering it with the inverse filter obtained by solving the autocorrelation normal equations for the first four filter coefficients (A's). The error signal should then have much of the formant structure removed, simplifying the identification of pitch. The pitch period estimates and voicing decision are made by peak picking on the autocorrelation sequence of the error signal.

The next steps in the pitch analysis, therefore, are identical to those used in the coefficient analysis. The first five autocorrelation coefficients of the windowed data are computed using AUTOCORR. The autocorrelation normal equations are then solved for four inverse filter coefficients using SOLVE. The time required for these steps is 380 microseconds.

The inverse filter algorithm is a direct five-term filter followed by a low-pass filter $1/(1-C\cdot Z^{-1})$. The inverse filter equation used is:

$$y_i = W_i + \sum_{j=1}^{4} A_j W_{i-j} \qquad i=0,\cdots,85$$

where $W_i=0$ for $i<0$.

The low pass filter is incorporated into the inverse filter calculation to produce the composition:

$$y_i = (W_i + \sum_{j=1}^{4} A_j W_{i-j})(1-C)+C\, y_{i-1}, \quad i=0,\cdots,85$$

$$= \sum_{j=1}^{4} B_j W_{i-j} + C y_{i-j}, \quad i=0,\cdots,85$$

where $B_j = A_j(1-C)$, $B_o = (1-C)$ and $y_{-1} = 0$.

The implementation of this equation requires only six multiplies and five additions per point. The preparation required is to convert the four filter coefficients to 16-bit rounded values and multiply them by 1-C. The total filter calculation requires 325 microseconds in program INVFIL.

Before calculating the autocorrelation sequence of the error signal, the mean of the error signal is computed and subtracted from it. If the data being autocorrelated did not have zero mean, the autocorrelation of the bias would make proper peak detection difficult by increasing the correlation of the shorter lag coefficients relative to those to the right of them. [2] The calculation of the mean and its subtraction is straightforward and takes only 75 microseconds in AP program ZMEAN.

The calculation of the autocorrelation sequence of the error signal is computationally the most significant task in the analysis. Autocorrelation coefficients must be calculated with time displacements covering the entire range of pitch intervals to be detected. For the initial experiments, the range of pitch detectable is about 50 to 360Hz, requiring pitch intervals of 2.7 to 20 milliseconds or pitch lengths at 450 microseconds per point of six to 44 points. The actual computation

includes all autocorrelation coefficients from 0 to 44, although only R_6 to R_{44} are tested for peak location. Since only 30 cells of data pad are available for use as accumulators in the autocorrelation program, two passes are used. During the first pass, R_0 through R_{24} are computed and stored. During the second pass, the remaining terms are computed. The autocorrelation coefficients are stored as 32-bit integers for peak picking. The pitch autocorrelation requires 1,910 microseconds in the AP.

The pitch estimate is made by locating the peak in the autocorrelation sequence, ignoring those points which correspond to pitches shorter than 2.7 milliseconds. Once a preliminary peak is located, a second search is performed to determine if a peak at twice the pitch length was selected. This search determines if an autocorrelation coefficient with index less than 3/4 of that of the peak has a value greater than 85% of the maximum. If so, the earlier peak is taken as the pitch estimate. The peak picking is performed directly on the unnormalized 32-bit autocorrelation coefficients. The time required in AP program PITCH is about 60 microseconds.

To increase the resolution of the pitch estimate, parabolic interpolation about the peak location is used. The interpolated correlation value is then normalized by division by R_0. One special case must also be tested for. If the peak value is the first in the range tested, it is possible that the value to its left is larger. In that case, a blind parabolic interpolation can yield a minimum rather than a maximum. To avoid this, if the value to the left of the peak location is greater than the peak value, the frame is declared unvoiced by forcing the correlation to 0. The algorithm for parabolic interpolation is:

Let R_p be the peak autocorrelation coefficient.

1. Compute $A = .5(R_{p-1}+R_{p+1}-2R_p)$

   $B = .25(R_{p-1}-R_{p+1})$

2. Convert R_0, R_p, A and B to 24-bit scaled values, compute R_0^{-1} and A^{-1}

3. $P = p+BA^{-1}$

$R_p' = R_p - B^2A^{-1}$

4. $V = R_p' R_0^{-1}$

The results 256P and V are converted to 16 bits and stored with the gain and K parameters from the coefficient analysis. The array processor program PEAKFIT performs the pitch interpolation in 25 microseconds.

The array processor portion of the analysis is complete at this stage. The remaining pitch and voicing decisions are largely logical operations, and are performed by the MP as is the encoding process. The total analysis program is too large to be loaded at one time into program store in the AP, so it is split into a root segment which is always present and two overlays. The majority of the calculation, through the final autocorrelation for pitch analysis, forms one overlay, and requires about 4,810 microseconds to run. The second overlay includes the peak location and pitch interpolation and takes 95 microseconds. The overlay transfer is managed by an interrupt process in the MP and takes about 20 microseconds.

When the array processor has completed its analysis calculation, the pitch position and correlation, gain and K parameters are input by the MP. A sequence of simple rules requiring the pitch estimates from the three preceding frames is used to refine the pitch and decide if the new frame is voiced or unvoiced. These decision rules are identical to those proposed by Markel. [4].

Let N1, N2 and N3 be the three previous frame pitches. Then the pitch N0 of the new frame is 3P if V>.4 or N1>0 and V>.3; otherwise, N0 is set to 0 to indicate unvoiced. The following rules are used to smooth the pitch and weight unvoiced-to-voiced transitions for the voiced pitch.

Rule 1 (smoothing): If $|N3-N1|<0.4N3$, replace N2 by the average of N1 and N3.

Rule 2 (extrapolation): If N3 is unvoiced, N2 is voiced and $|N1-N0|<0.2N1$, then replace N2 by 2 N1-N0.

The gain is also multiplied by 1.625 at this stage to partially correct for the loss in energy of the analysis data due to Hanning windowing.

One additional test is used to eliminate certain gross errors in the voicing decision. If the first K parameter $K_1 = -R_1$ is positive, it indicates that an extremely high frequency component is present. If this component is large, the signal could not be voiced. Hence, a test for $K_1 > .5$ is included after the pitch N2 is associated with buffer 3. If the test is passed, the frame is declared unvoiced.

In order to associate the pitch with the proper set of coefficient analysis parameters, we must shift it one 9.6 millisecond frame forward. This lines up the center of the pitch window and coefficient window. Thus, the final pitch N2 belongs with the gain and K parameters calculated with N3. Four one-parcel parameter buffers are used to hold this data until it is ready to encode.

The initial network voice protocol provides that data will be transmitted only every 19.2 milliseconds. Since the analysis frame rate is 9.6 milliseconds, only every other frame is selected for encoding and transmission. However, informal listening tests have convinced us that a simple selection rule can be used to improve the vocoder quality during periods of sharp change in gain. Since the gain parameter reflects not only the original signal energy, but also the extent to which the K parameters fail to describe the signal, large changes in gain reflect significant transitions in the signal. These transitions often occur much more rapidly than the 19.2 milliseconds for which each set of parameters transmitted must suffice. We have attempted to select the best available set of parameters to transmit from each three adjacent frames, rather than selecting every other set automatically. In order to do this, we buffer two additional parcels before transmission. When a parcel is to be transmitted, two tests are made:

Let G5,G4,G3 be the gains of the three parcels
If G4>3•G3+8, replace parcel 3 by parcel 4.
If G3>3•G5+8, replace parcel 5 by parcel 4.

Parcel five is then selected for transmission.

All parcels and pitch save values are advanced by one position each time, whether or not any parcel is needed for transmission.

That is

N0→N1→N2→N3 and

Parcel 0 → Parcel 1 → Parcel 2 → Parcel 3 → Parcel 4 → Parcel 5.

The encoding of a parcel for transmission is defined by a set of tables stored in the MP memory. There are five tables, one for pitch, one for gain, and three for seven-bit, six-bit, and five-bit codes for the K parameters. The tables used for both encoding and decoding are given in Table 1-1. The encoding is performed by a binary search technique to find the index to be transmitted. Packing of the codes into a message for network transmission is performed separately and is discussed in the next section.

1.1.3 THE RECEIVER SYSTEM

The receiver has a somewhat simplified job of data management compared with the transmitter. No overlapping windows are used, nor are there separate pitch and coefficient sections.

The synthesizer, itself, is a filter with periodically updated coefficients, excited by a train of pulses separated by the current pitch period for voiced frames, or a random noise generator for unvoiced frames. A pitch synchronous updating procedure is followed. The transmitted parameters are associated with the ends of the 19.2 msec frames. A new set of transmitted parameters is provided for the synthesizer each time a frame's worth of data is to be synthesized. These new parameters are associated with the end of the frame. The previous set is used for the beginning of the frame. The actual parameters used in the filter, and the gain, are then calculated by linear interpolation between the sets at each end. Exceptions are when the two sets are not both voiced or both unvoiced, in which cases the beginning parameters are used directly. Once a pitch period is started, the synthesizer filter is not altered until the end of the pitch period, regardless of any frame boundary crossings.

The MP processing for synthesis, itself, is limited to decoding the parameters for a frame and passing them to the AP, where the actual synthesis takes place. The decoding uses table indexing with the codes received in the companions to the five encoding tables. These values are also given in Table 1-1.

Table 1-1. Encoding and Decoding Tables

a) Pitch Parameter Coding

Encode Test Value	Code	Quantized Value	Encode Test Value	Code	Quantized Value
0	0	0	46	32	45
14	1	14	48	33	47
15	2	15	50	34	49
16	3	16	52	35	51
17	4	17	54	36	53
18	5	18	56	37	55
19	6	19	58	38	57
20	7	20	60	39	59
21	8	21	62	40	61
22	9	22	64	41	63
23	10	23	66	42	65
24	11	24	68	43	67
25	12	25	70	44	69
26	13	26	72	45	71
27	14	27	74	46	73
28	15	28	76	47	75
29	16	29	78	48	77
30	17	30	81	49	80
31	18	31	84	50	83
32	19	32	87	51	86
33	20	33	90	52	89
34	21	34	93	53	92
35	22	35	96	54	95
36	23	36	99	55	98
37	24	37	103	56	101
38	25	38	107	57	105
39	26	39	111	58	109
40	27	40	115	59	113
41	28	41	119	60	117
42	29	42	124	61	122
43	30	43	129	62	127
44	31	44	32767	63	133

Table 1-1 (continued)

b) Gain Parameter Coding

Encode Test Value	Code	Quantized Value	Encode Test Value	Code	Quantized Value
15	0	0	137	16	128
17	1	16	158	17	147
20	2	18	181	18	169
23	3	21	208	19	194
26	4	24	239	20	223
30	5	28	274	21	256
34	6	32	315	22	294
39	7	37	362	23	338
45	8	42	416	24	388
52	9	49	478	25	446
60	10	56	549	26	512
69	11	64	630	27	588
79	12	74	724	28	676
91	13	84	832	29	776
104	14	97	955	30	891
119	15	111	32767	31	1024

Table 1-1 (continued)

c) Seven-Bit K Parameters

Encode Test Value	Code	Quantized Value	Encode Test Value	Code	Quantized Value
402	0	0	23453	32	23170
1206	1	804	24008	33	23732
2009	2	1608	24548	34	24279
2811	3	2411	25073	35	24812
3612	4	3212	35583	36	25330
4410	5	4011	26078	37	25833
5205	6	4808	26557	38	26320
5998	7	5602	27020	39	26791
6787	8	6393	27467	40	27246
7571	9	7180	27897	41	27684
8351	10	7962	28311	42	28106
9127	11	8740	28707	43	28511
9596	12	9512	29086	44	28899
10660	13	10279	29448	45	29269
11417	14	11039	29792	46	29622
12167	15	11793	30118	47	29957
12910	16	12540	30425	48	30274
13646	17	13279	30715	49	30572
14373	18	14010	30986	50	30853
15091	19	14733	31238	51	31114
15800	20	15447	31471	52	31357
16500	21	16151	31686	53	31581
17190	22	16846	31881	54	31786
17869	23	17531	32058	55	31972
18538	24	18205	32214	56	32138
19195	25	18868	32352	57	32286
19841	26	19520	32470	58	32413
20475	27	20160	32568	59	32522
21097	28	20788	32647	60	32610
21706	29	21403	32706	61	32679
22302	30	22006	32746	62	32729
22884	31	22595	----	63	32753

Table 1-1 (continued)

d) Six-Bit K Parameters

Encode Test Value	Code	Quantized Value	Encode Test Value	Code	Quantized Value
804	0	0	23732	16	23170
2411	1	1608	24812	17	24279
4011	2	3212	25833	18	25330
5602	3	4808	26791	19	26320
7180	4	6393	27684	20	27246
8740	5	7962	28511	21	28106
10279	6	9512	29269	22	28899
11793	7	11039	29957	23	29622
13279	8	12540	30572	24	30274
14733	9	14010	31114	25	30853
16151	10	15447	31581	26	31357
17531	11	16846	31972	27	31786
18868	12	18205	32286	28	32138
20160	13	19520	32522	29	32413
21403	14	20788	32679	30	32610
22595	15	22006	-----	31	32729

e) Five-Bit K Parameters

1608	0	0	24279	8	23170
4808	1	3212	26320	9	25330
7962	2	6393	28106	10	27246
11039	3	9512	29622	11	28899
14010	4	12540	30853	12	30274
16846	5	15447	31786	13	31357
19520	6	18205	32413	14	32138
22006	7	20788	-----	15	32610

The AP synthesis processing is primarily involved with the synthesis filter, itself, so we will examine this program first. The only other major program component is the parameter update routine, which is used at pitch boundaries.

The synthesis filter is a two-multiplier lattice filter which uses the K parameters directly. Figure 1-5 is a diagram of the lattice structure of this filter. The algorithmic implementation of this filter is:

1. $Y = DRV - K_{10}B_{10}$

2. FOR $j=9,\cdots,1$

$$Y = Y - K_jB_j$$

$$B_{j+1} = B_j + KY$$

3. $B_1 = Y$

$$Y = GY$$

The B's represent the filter memory. This memory is initially 0, and is updated as shown during each pass through the filter. The factor G is the transmitted gain factor adjusted for the length of the pitch period. The gain is used as a post multiplier to permit scaling of the input value DRV as large as possible to preserve maximum accuracy with 16-bit rounded arithmetic. However, since the filter, itself, does have resonances, care must be taken to avoid overflow in the intermediate computations. The input driving function amplitudes were chosen experimentally to avoid overflow as

$$DRV = \begin{cases} 2^{-7}\left(1,\frac{-1}{P-1},\frac{-1}{P-1},\cdots,\frac{-1}{P-1}\right) & \text{for voiced periods} \\ r_n, \text{ the output of a uniform random number generator,} & \text{for unvoiced periods} \end{cases}$$

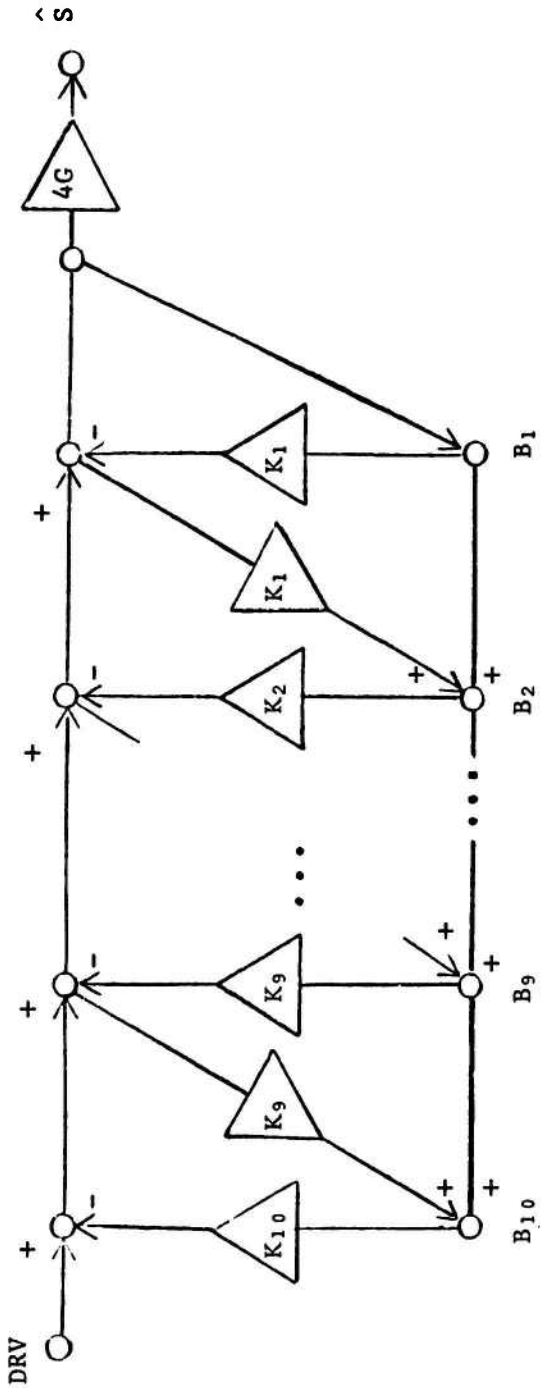


Figure 1-5. Lattice Filter Structure

The average energy of each pseudo random value rn is $3^{-1/2} \cdot 2^{-10}$. The energy of the output signal is then adjusted by multplication by a gain factor

$$G = \begin{cases} 8\sigma\sqrt{2P} & \text{for voiced periods} \\ 16\sigma\sqrt{6} & \text{for unvoiced periods} \end{cases}$$

σ is the interpolated gain parameter. Unvoiced periods are 128 points long, as is the analysis frame size. The filter is implemented as a subroutine ISYN requiring 9.5 microseconds per point. This subroutine is described in detail in the appendix.

Since the analysis input data was pre-emphasized by a filter $1-\mu Z^{-1}$, the inverse of that filter must be used for post emphasis. The implementation of the post emphasis filter is:

$$\hat{S}_n = Y+\mu S_{n-1}$$

For unvoiced periods, a pseudo random number generator is used to provide the driving function DRV. A multiplicative congruential generator with multiplier $\eta = 125$ is used

$$x = (125x \mid \text{modulo} \pm 1)$$
$$r_n = 2^{-10}x$$

Subroutine RANDOM requires 1.5 microseconds to compute each random number.

The synthesis program for each pitch period can now be described completely. The flowchart for its operation is given in Figure 1-6. The value NEG is $-2^{-7}/(p-1)$, a negative bias term to cancel the delta function at the start of the pitch period. When a pitch period is complete, NEWPITCH computes the new filter parameters and driving function. When a buffer is full, the program SYNOUT interrupts the MP and transfers the buffer. Next time the synthesis program is executed, processing resumes at SYN for a new buffer. Execution time is 1495 microseconds for a 128 point buffer of voiced data and 1685 microseconds for a buffer of unvoiced data.

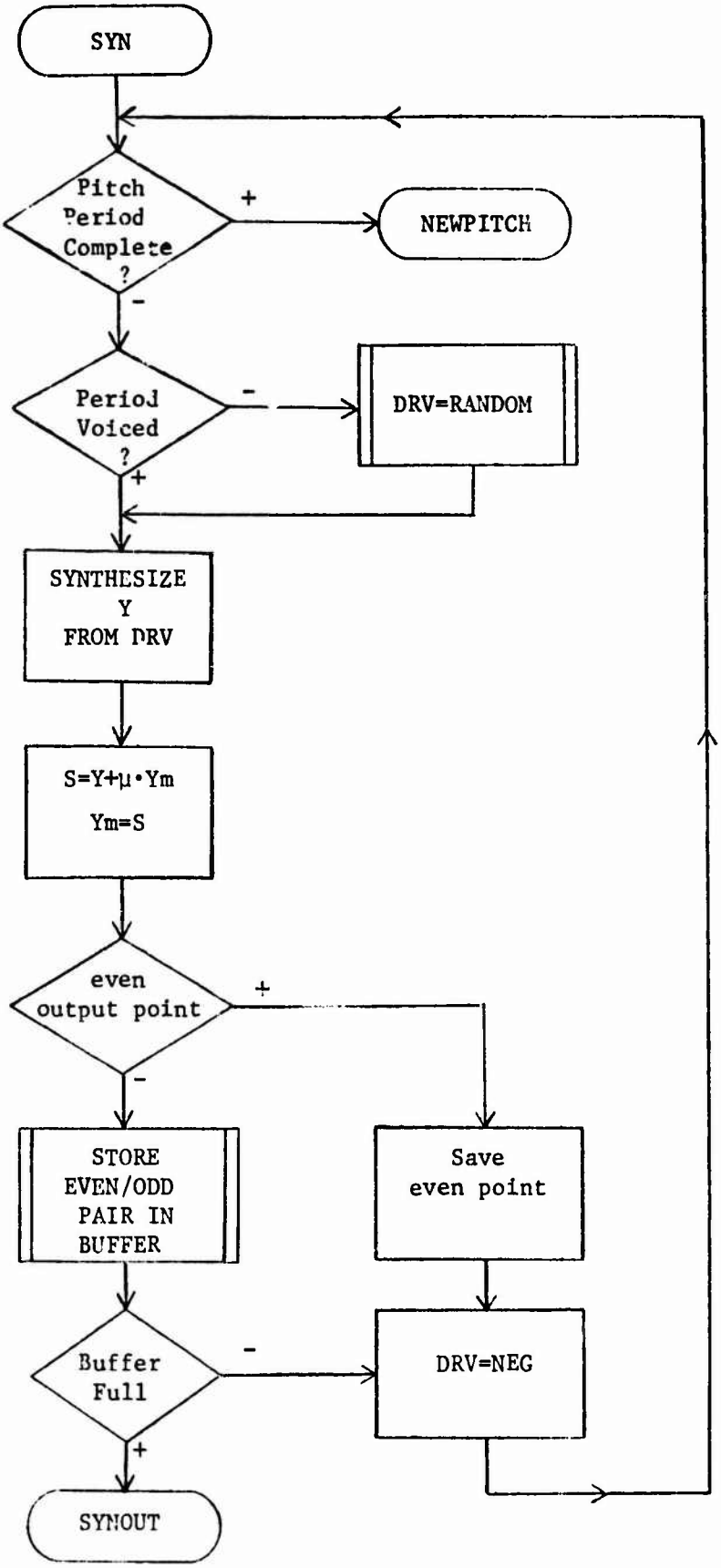


Figure 1-6. Synthesis Filter Flowchart

Each time a pitch period is complete the parameters used are updated. This updating is performed by using the position of the next output point in the buffer to interpolate between the parameters associated with the left and right ends of the block. No interpolation is used, however, if the two sets of parameters are not both voiced or both unvoiced. In these cases, the left frame parameters are used directly. The interpolation value is $\alpha=i/128$, where i is the index of the next output point, $i=0,\cdots,127$. The K parameters, pitch and gain values are all interpolated by the linear interpolation formulae:

$$P = P_L - \alpha P_L + \alpha P_R$$

$$g = g_L - \alpha g_L + \alpha g_R$$

$$K(i) = K_L(i) - \alpha K_L(i) + \alpha K_R(i)$$

If the new pitch period is to be unvoiced, the gain value is used directly as the post multiplier G in the filter (all gain values are adjusted by either $16\sqrt{6}$ or $8\sqrt{2}$ when they are first received). The period p is 128 points for an unvoiced period.

For voiced periods it is necessary to calculate the gain adjustment $\sqrt{p}$ and the appropriate delta function compensation $(p-1)^{-1}$. The interpolated gain g is multiplied by $\sqrt{p}$, NEG is set to $2^{-7}(p-1)^{-1}$, and DRV is initialized to 2^{-7}.

The interpolation program NEWPITCH exits to SYN, where the synthesis filtering resumes. The filter memory (B_1 to B_{10}) is not changed during the interpolation. The computation time if interpolation is needed is 35 microseconds for voiced periods and 20 microseconds for unvoiced periods.

As has been mentioned earlier, the beginning of a new buffer of 128 points has no immediate effect on the filter coefficients. A new parcel of parameters is received from the MP at this time. The gain parameter is immediately multiplied by $16\sqrt{6}$ if the parcel is unvoiced or $8\sqrt{2}$ if the parcel is voiced. The old right parcel parameters are moved

into the left parcel buffer and the new parameters are stored in the right parcel buffer. Processing then begins at SYN. These initializing steps are performed in program INITSYN and take 26 microseconds.

The synthesis program in the array processor fits into the same overlay region of program store as the analysis overlays. It is loaded by the MP and one parcel of decoded parameters is transferred to it. Thereafter, only the AP is involved until a complete block of 128 points has been synthesized. This block is input to an output data buffer in the MP by an interrupt process. The total AP time for each block is 1700 to 1900 microseconds depending on whether voiced or unvoiced data is produced and how many new pitch periods start.

1.1.4 DIFFERENCES IN CHI IMPLEMENTATION

Although we have closely followed the work of Markel in defining the implementation of both the analysis and synthesis portions of the speech compression system, we have chosen to make some changes. These changes were in some cases dictated by differences in machine architecture between the CHI AP-90 and the SPS-41 which most NSC group members are using. In other cases, they represent what we believe are either corrections of errors in the described implementation or worthwhile improvements. All these changes were described in the text of the transmitter and receiver portions of this section. They are summarized here for easy identification.

Analysis

1. Rounded arithmetic is used for all multiplication and scaling except in the autocorrelation program where a full double precision product is kept. The rounding is symmetric about zero, so no negative bias is introduced.

2. Input data is not pre-scaled.

3. A Hanning rather than Hamming window is used, since this allows us to use the cosine table in read-only memory without any extra multiplications.

4. The two-multiplier lattice filter is used for the digital low pass filtering of input data before downsampling.

5. Normalized autocorrelation coefficients are used for solving the normal equations.

6. 24-bit scaled arithmetic is used for all computations in the normal equations solution.

7. The mean is extracted from the error signal before autocorrelation.

8. We require that the peak selected in the pitch analysis be a local maximum to avoid false peaks at the extremes.

9. The output parcel is declared unvoiced if $K_1 > .5$.

10. Parcel selection is used to provide better representation of the signal in regions of rapid change.

Synthesis

1. Rounded arithmetic is used throughout the calculation with the exception of the random number generator.

2. The interpolation coefficient is defined as the output index i divided by the blocksize, not the blocksize -1.

3. The interpolation formula is implemented as $K(i) = \left(K_L(i) - \alpha K_L(i)\right) + \alpha K_R(i)$ instead of $K(i) = \left(K_R(i) - K_L(i)\right) \alpha + K_L(i)$, since the quantity $K_R(i) - K_L(i)$ can overflow in fixed point computation.

4. A two-multiplier lattice filter was used for synthesis.

1.1. APPENDIX

The program timings show that out of approximately five milliseconds required for coefficient analysis, over three milliseconds are spent in autocorrelation calculations. For synthesis, the filter calculation uses about 1.5 milliseconds out of less than 2 milliseconds total time. It is clear, then, that the autocorrelation and lattice filter calculations are critical. We shall go through the AP micro code for the inner loops of each of these in some detail.

Autocorrelation

The computation is performed in double precision fixed point with 16-bit input. The input data is packed two entries per 32-bit word in MOS memory. Data pad cells 32-M through 31 are used to accumulate the autocorrelation values for up to 29 coefficients. During each pass, a pair of entries x_i and x_{i+1} are fetched from memory and stored in DP_o, data is then fetched from memory starting at x_{i+j}, where R_j is the first autocorrelation coefficient being computed (j is even). The sequence of micro operations for each multiply-add will be shown with time running

vertically and micro operations occurring in parallel shown on the same line; all parallel operations change their destinations at the same time.

M1 = Multiplier, M2 = Multiplicand, AC = Accumulator,
C = 32 bit register, DPA = data pad address register
DP = current data pad cell. MA = memory address register
MD = memory data register, SP(n) = Index register n
MP = product.

Initially:

MD = x_{i+j},x_{i+j+1}, M1=x_i, M2=x_j,MA=A(x_{i+j})

DPA = 31-M. Start at instruction 1.

	R_{n-1}	R_n	R_{n+1}
0	DP+AC→AC	(Multiply)	---
1	AC→DP INC DPA	MP→C	MD→AC INC MA(start fetch)
2	IF DPA=31 GOTO7	C→AC	ACR→M2(Start Multiply)
3	DP+AC→AC	(Multiply)	(Memory Fetch)
4	AC→DP INC DPA	MP→C	(Memory Fetch)
5	If DPA≠31 GOTO 0	C→AC	MDL→M2(Start Multiply)
6	Advance to next pair x_i,x_{i+1}, IF NOT DONE, BEGIN AT 1 AGAIN.		
7 & 8	Start Memory fetch for x_{i+j}, x_{i+j+1}		
9	DP+AC→AC		
10	AC→DP INC DPA		
11	GOTO 3	DPR→M1 MDR→M2 (Start multiply)	INC MA DPA=31-M (Start fetch)

Up to three separate coefficients are being calculated at any one time, reducing the total time per calculation of six clocks or 1 microsecond to an effective time of three clocks per multiply add.

Two-Multiplier Lattice Filter (ISYN)

This filter is used for both synthesis and the digital low pass filter prior to downsampling for pitch analysis. The K parameters are stored in reverse order in the left half of consecutive data pad cells. The filter memory elements are stored in the right half of the same data pad cells. The program begins with DRV in ACL and G in DP_0R. Comments are in ( ).

```
0  DPA=32-M (j=M)
1  M1 = DPL M2 = DPR (start K_M·B_M)
2  2→SP10
3  MP→ACR GOTO 7
4  MP→ACR (Y·K_j) DEC DPA(j) M1 = DPL, M2 = DPR (start K_j-1·B_j-1)
5  DPR+ACR→ACR(B_j+Y·K_j)     DEC DPA(j+1) (MULTIPLY)
6  ACR→DPR (Update B_j+1)     MP→ACR(K_j-1·B_j-1) DPA+SP0→ DPA(j=j-1)
7  M2=ACL-ACR(Start Y·K_j)    ACL-ACR→ACL (New Y)
8  If DPA≠31 GOTO 4           INC DPA(j+1)
9  MP→ACR(Y·K_1)              DEC DPA(31) M1=DPR (start Y·G)
10 DPR+ACR→ACR (B_1+Y·K_1 )   DEC DPA(30)
11 ACR→DPR (Update B_2)       ACL→ACR INC DPA(31)
12 ACR→DPR (B_1=Y)            MP→ACR(Y)
13 EXIT
```

For this filter, the time for each section is reduced from eight clocks to five through parallelism. Thus, the effective rate is .833 microseconds/coefficient.

Table 1-2. AP Program Summary

N = Analysis Window Size

S = Downsampled Pitch Window

D = Downsampling Factor

M = Number of K's

	Size	Adds	Multiplies	Time(μsec)
Analysis				
Data Input (N)	8	N	0	1.67N+1
DOWN(N,D)	60	5N	5N	$\frac{N}{6}\frac{5}{D}$ + 39)+28
DIFFWIN(N)	31	N	2N	1.67N
AUTOCORR (M+1,N)	32	(M+1)N	(M+1)(N+1)	$\frac{M}{2}$(N+5)+N+8
SOLVE(M)	101	M^2+.5M	M^2+6.5M	$1.5M^2$+10M-5
FIXK(M)	21	M+5	6	1.5M+10
AUTOCORR(5,S)	8	5S	5S+5	3.5S+20
SOLVE(4)	5	18	29	60
INVFIL(S)	45	6S	6S	3.33S+10
ZMEAN(S)	15	2S	1	.833S
PITCHCORR(44,S)	16	46S	46S	25S-240
PITCH(S)	37	S+$\frac{S}{2}$-10	0	S+3
PEAKFIT	51	9	6	23
OUTDATA(M)	5	0	0	.667(M+2)
Synthesis				
INITSYN	47	0	1	1.5M+11
SYN	32	(2M+1)N	(2M+1)N	.866N(M+4)
SYNOUT	14	0	0	.5M+.667N+4
NEWPITCH	71	M+2	2M+4	1.33M+24
RANDOM	8	0	1	1.33

1.2 SYSTEM ASPECTS OF SPEECH COMPRESSION ON THE ARPANET

We have described the transmitter and receiver systems as they operate on speech data to produce encoded parcels and as they take encoded parcels and synthesize speech data for output. In this section, we shall discuss the system which surrounds and uses these programs.

Actually, several different systems have been developed. During the implementation and checkout of the LPC programs it was necessary to provide a non-real-time system using pre-digitized data recorded on disk for analysis and producing its synthesis output on disk. This system permits examination of any of the intermediate results and proved invaluable in discovering the causes of any difficulties as they occurred. It also permits experimentation with the effects of different modifications in the algorithms on identical input data. This system will not be discussed further here since all systems we have developed can use the same array processor programs and our primary interest is the use of LPC on the network.

The real time LPC system on the network is organized as a set of cooperating processes running on three different processors. These processes must input and output analog data, receive and send control and data messages on the network according to the Network Voice Protocol and interpret local control from the keyboard in addition to the LPC functions already discussed. Figure 1-7 shows how these processes cooperate to pass information through the system.

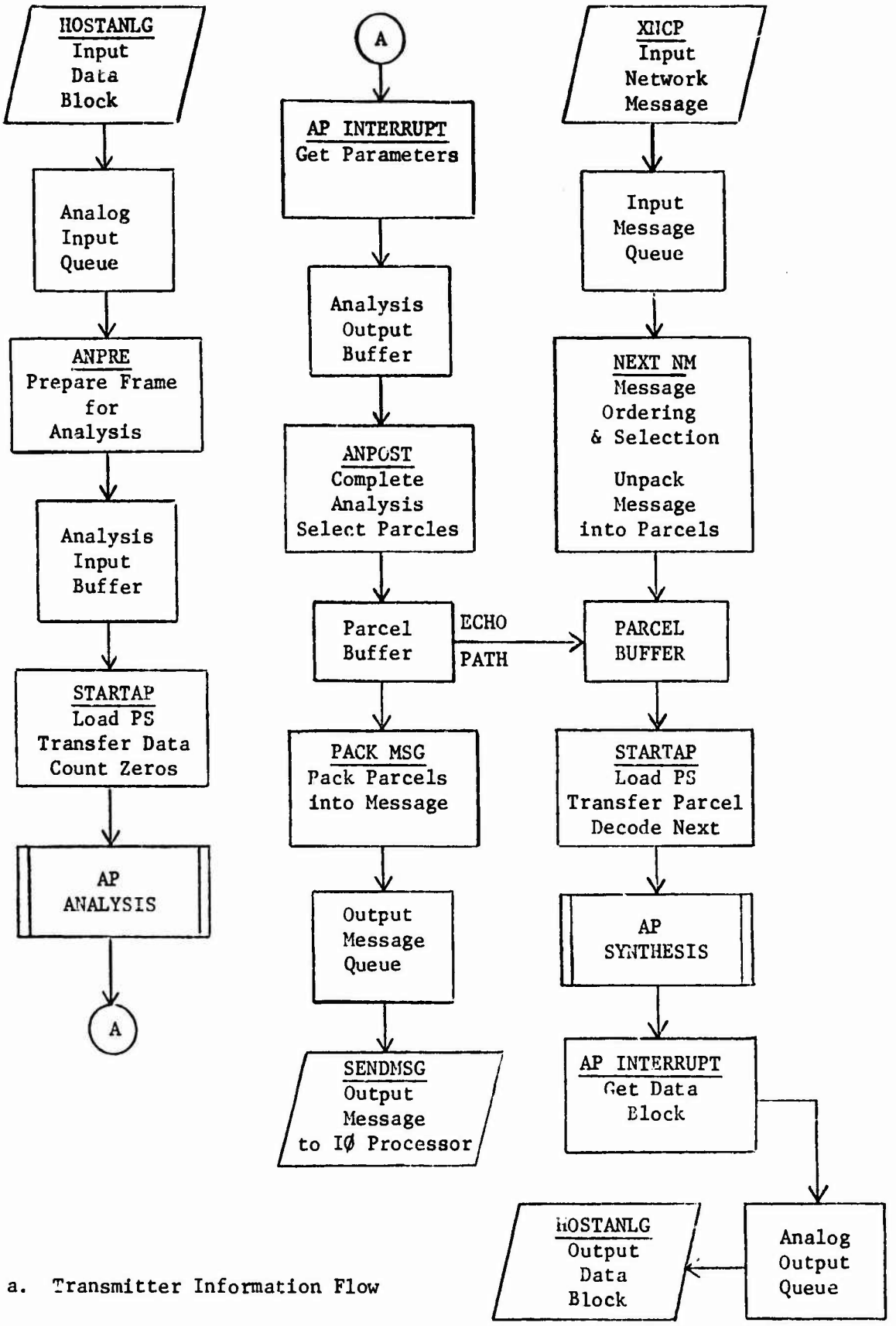
1.2.1 I/O PROCESSOR

In our current system all external input and output is handled by a separate processor. This processor is similar to the MP-32A in functional capability . We will refer to it as the I/O processor. Its responsibilities are:

1. Sampling analog input data, converting it to 12-bit two's complement numbers, and buffering these numbers in 128 word blocks.

2. Converting digital data to analog during output from blocks of 128 words.

3. Transferring a buffer to the MP-32A for processing and receiving a buffer of output data (HSTANLG).



a. Transmitter Information Flow

b. Receiver Information Flow

Figure 1-7

4. Implementing the reliable transmission program (RTP) required for a very distant host connection to the ARPA network.

5. Supporting the XNCP output functions to send NVP messages on the network and signalling the MP when all messages have been sent (RDY).

6. Receiving messages and interpreting message type codes on XNCP links. Buffering these messages and transmitting them to the MP if types 0 or 3 (XNCP) or types 5 or 9 (SXNCP).

1.2.2 MP INTERRUPT PROCESSES

The MP responds to four separate interrupts from the I/O processor. In addition, one non-interrupt process in the MP communicates with the I/O processor.

The analog I/O interrupt (HSTANLG) from the I/O processor signals that a block of speech data is complete. This process obtains an analog data buffer, inputs the data and places the buffer on the analog input queue (AIQ). It then removes a buffer from the analog output queue (AOQ) and transfers its data to the I/O processor for conversion. If the analog output queue is empty, a block of zeros is sent instead and the receiver pseudo timer (CTIME) is advanced by 19 milliseconds.

The network messages interrupt (XNCP) from the I/O processor signals that a regular or uncontrolled message has been received. A message buffer is obtained and the message is input into it. The length of the message and the time the message was input are also recorded in the buffer. The link field of the IMP-HOST leader is tested for the data link or other and the buffer is linked to either the input queue for synthesis (SINQ) or the control message queue (CTRLQ).

The special message interrupt (SXNCP) signals the arrival of a non-interesting message (not type 0 or 3). Two 16-bit words are input; if the message is not a RFNM, it is placed on the control queue for later analysis.

The output queue empty interrupt (RDY) is sent by the I/O processor to indicate that all messages have been sent to the IMP successfully. It is used to allow the next message to be sent to the I/O processor when uncontrolled type messages are used on the network. This signal allows the MP to do all output message buffering. The interrupt routine only enables output.

The output message process SENDMSG is scheduled whenever the output message queue (OMSGQ) is not empty. If output is enabled and the receiver is ready (as indicated by control message 6), the I/O processor is interrupted and the message is removed from the queue and output to it. Otherwise, the process terminates and is rescheduled later.

The array processor processes have already been discussed. Scheduling of the AP processes is handled by the MP process STARTAP. AP processes end by interrupting the MP. Three interrupt processes are involved. Two input the results of the analysis or synthesis programs. The third loads the necessary program store overlay in order for the AP to complete the analysis.

The interrupt signalling completion of the analysis causes the MP to input the pitch position, its correlation and the coefficient analysis parameters. A fixed buffer in MP-MOS is used for these quantities. The interrupt routine marks the completion of the AP portion of the analysis and marks the AP as free.

When a synthesis complete interrupt is received, a data buffer is allocated and the synthesized data from the AP transferred into it. This buffer is enqued on the analog output queue for transfer to the I/O processor. The AP is marked as free.

1.2.3 SCHEDULED PROCESSES

The array processor scheduling is an important part of the responsibilities of the MP. Even at a 150 microsecond sampling rate, the AP processes need about 13 milliseconds of each 19.2 millisecond frame time. If additional computations are needed for multiple speaker synthesis or other new tasks, even less idle time will be available. The scheduler must also select between analysis and synthesis in such a way as to assure a steady stream of data for analog output and parcels for transmission. The array processor scheduler (STARTAP) is an MP process which is activated whenever the AP is marked free. The priority scheme it uses to choose between analysis and synthesis is the following:

1. First priority is given to synthesis if the analog output queue is empty and a parcel of parameters is available.

2. Second priority is given to analysis, provided that a block of data has been prepared (see ANPRE) and the previous analysis output has

been processed out of the fixed analysis output buffer.

3. Lowest priority is given to synthesis if the analog output queue is not empty. In no case will more than a fixed number (currently 5) blocks of analog output be allowed to appear on the queue. This restriction is to avoid monopolization of data buffers by the output data.

The preference for analysis over synthesis allows a minimization of the number of data buffers required, taking advantage of the data compression to be able to hold more information in less memory space.

The initiation of an analysis process includes the calculation of the number of zero crossings in the data as it is output to the AP. The zero crossing count is sent as the last word to the AP.

When a synthesis process is started, a set of previously decoded parameters must be passed to the AP. Immediately after starting the AP process, the parcel pointer is advanced and the next parcel in the current message, if any, is decoded. The receiver pseudo timer, CTIME, used in message selection, is updated by 19 milliseconds.

The data buffers on the analog input queue each hold 19.2 milliseconds worth of data, but analysis is performed on 19.2 milliseconds worth of data every 9.6 milliseconds. Each buffer, therefore, contains data which will be used in three different analysis passes. The buffer must be split into two parts, and the parts properly sequenced for AP processing. A separate MP process (ANPRE) is responsible for this data management. This process is scheduled each time an analysis is initiated. It prepares a fixed analysis input buffer for the next pass by moving the second half of the buffer into the first half then either getting a new buffer from the analog input queue and moving the first half of the new buffer into the second half of the analysis input buffer or moving the second half of a previously located buffer in and releasing that buffer. It then marks the analysis input buffer ready.

When the array processor has completed an analysis pass the MP post analysis process (ANPOST) is scheduled. The first part of this process was discussed in the previous section. The preliminary voicing decision is made and refinement of the pitch estimate occurs. The analysis output parameters are processed out of the output buffer. On every other pass

a parcel is selected for transmission, encoded and placed in an output list until sufficient parcels have been collected to pack into a message. At the same time, a test is made of the gain of the parcel. If the gain is below a threshold (SLNGTH) for a specified number (TBS) of consecutive parcels, silence is declared and any further parcels whose gain is below threshold are not sent. The logic for these decisions is illustrated in section 1.3 during the discussion of message ordering.

Once sufficient parcels are collected, the output message preparation process (PACKMSG) is scheduled to pack these parcels into a message. The number of parcels in each message is intended to vary from a nominal minimum to an absolute maximum, which for uncontrolled messages must be such that the message is only one packet long. To achieve this variation, whose purpose is to allow more efficient transmission when a backlog exists, but minimize delays when traffic is flowing, several criteria exist for scheduling this process. Normally, it is scheduled when the number of parcels available exceeds the specified minimum, and the output message queue is empty,and the I/O processor has signalled that it is ready, and the receiver is believed to be ready. However, if the analysis post processing routine prepares the maximum allowed parcels, it immediately invokes the message preparation process. Additionally, the message preparation process is invoked to send the parcels which are ready when silence is delcared.

The message preparation process obtains a message buffer, uses MTIME as the time stamp for the message and packs the parcels into the buffer, adding a padding count as the lest five bits of the message. The HOST-IMP leader is moved into the message and the four-bit sequence field in the leader source is updated for the next message. The buffer is then placed on the output message queue with the message length and time of preparation also recorded in the buffer. The unpacked parcels can also be transferred directly to the receiver for synthesis if silence is being received and the echo option is enabled.

When all parcels in the current message being processed by the receiver are exhausted, the input message process (NEXTINM) is scheduled. Since the method and timing of message selection are described in detail

in the section on message ordering, they will not be discussed here. When a non-silence message is selected, it is unpacked into a list with each parcel occupying 12 words. The parameters are still encoded. The beginning and end of this list are marked with pointers and the first parcel is decoded. As each parcel is sent to the array processor for synthesis the front pointer is incremented to the next parcel and that parcel is decoded.

Local control over several parameters of the network speech compression system is supported through a keyboard input interpreter. This process is scheduled whenever input is received from the local keyboard. It allows the following operations:

1. Termination of the connection.

2. Enabling or disabling synthesis of output parcels for local playback during silence in the receiver (ECHO option).

3. Set the value for any of the following system parameters:
   a) Time Before Silence (TBS)
   b) Silence Threshold (SLNGTH)
   c) Time After Silence before Synthesis Starts (TAS)

4. Enable tracking of messages sent.

5. Write the current trace record onto disk. Tracing of all input messages is always performed if a trace file is provided when the network speech compression system is started. This trace is illustrated in Section 1.3.4.

NVP control messages are processed in the NVP control process. This process is scheduled whenever the control queue is not empty. During the establishment of a connection it can also be entered to start the caller sequence as defined in the NVP for a specified HOST. [9] The protocol section can function as either a called or an answerer, or both in the case of experiments in communication with ourselves over the net. Only a limited subset of the options specified in the NVP are supported during the initial connection. Once the connection is established, only INQ, READY, NOT READY and TERMINATE are recognized.

MP processes are assigned processor service by a simple priority cycle which examines each process in turn to see if all conditions

necessary for it to receive service are satisfied. Once started, each process is allowed to run to completion, except for interruption by interrupt processes. The cycler begins its scan with the highest priority process each time. The current priority order is the AP scheduler, followed by analysis post processing, analysis preprocessing and input message selection and preparation. Output message preparation is given lowest priority, but may be invoked as part of the analysis post processing as explained earlier.

We have adopted this cooperating processes approach for several reasons. First, a simple alternation of analysis and synthesis, or any fixed combination of them, is limiting, if only because no synthesis is needed during receiver silence, but analysis must occur at an average rate of 9.6 milliseconds. The separate process approach also permits flexibility in adding new processes for monitoring, conferencing or other purposes as well as moving processes from one processor to another if necessary and possible.

1.3 ORDERING OF NVP MESSAGES

Since standard HOST-to-HOST messages, even without the use of standard protocol, were found to introduce too much delay for comfortable interactive voice communication, a new network HOST-IMP and IMP-HOST message type was defined. This message type eliminates the normal message ordering and error control mechanisms in the IMP. As a result, messages may arrive out of order or not arrive at all (BBN Report No. 1822 revised 12/74).

The removal of IMP message control does speed up transmission of messages, but places a burden on the HOST-HOST protocol to provide enough information in the messages themselves to allow whatever re-ordering is needed, as well as adjusting to lost messages. The Network Voice Protocol (NVP), as defined in NSC Note 43, [9] provides for a four-bit sequence number and a 15-bit time stamp in each data message. This section presents the way that this information has been used to accomplish the message control functions. Whenever possible, we have tried to indicate the reasons we chose the method we did, and to indicate the pitfalls to be avoided. The development of these methods has been assisted by discussions with Jim Forgie of Lincoln Labs.

We begin this section with a description of the relevant features provided in the NVP and the system structure used to implement it. This is followed by a discussion of message control and the use of the message time stamp. Finally, we summarize our suggestions. The appendix presents some statistics gathered in using uncontrolled messages for network voice communication.

1.3.1 NETWORK VOICE PROTOCOL

At the transmitter, analog data is typically collected into blocks, with each block containing a fixed number of points and representing a fixed amount of time. For the present systems, this block size is 128 points representing 19.2 milliseconds of input data. A variable number of such blocks can be buffered pending analysis. The analysis produces a new set of parameters for each 9.6 millisecond period, although only one set of parameters is selected for transmission for each 19.2 milliseconds. Normally, from seven to 13 sets of parameters are encoded and packed into one message. This message carries a 15-bit time stamp giving the time,

in milliseconds, when the first data of this message was processed. However, the NVP provides for transmission of silence messages in lieu of the normal parameter transmission during long silence periods. A long silence is declared if the gain parameter from the analysis remains below a threshold value for more than a specified length of time. When long silence is declared, any parameters waiting to be transmitted may be sent. Thereafter, silence messages containing only the time stamp and a silence bit are sent at intervals of time-send-silence seconds. During long silence, analysis of input blocks continues. When the gain parameter exceeds the silence threshold, long silence is terminated. Parameters representing the last few blocks of data before the threshold was exceeded are then included in the first non-silence mssage transmitted.

When the receiver gets a data message from the network, it is buffered until the message being processed is exhausted. At that time, the next message in sequence is selected for processing. If the next message in sequence has not arrived, a decision must be made on whether to wait for it or to assume it is lost and process some other message. If a message arrives after its successor has already been processed, it must be discarded. In order to provide a steady supply of data for output, in the face of varying transmission times for messages over the network, a delay is provided from the time the first non-silence message after a period of long silence is received before the message is processed. This time after silence delay permits a queue of input messages to build up. Typically, the variation in transit times across the country (10-12 IMP hops) is around 0.5 second. A time-after-silence delay of 0.5 seconds results in a queue averaging three or four non-silence messages, if minimum length messages contain seven or eight parcels.

1.3.2 MESSAGE ORDERING

The use of uncontrolled (type 3) messages for speech data transmission requires the HOSTs to assume the burden of re-ordering message and discarding past-time, out-of-order messages as well as surviving lost messages. Actual experience with type 3 messages indicates that both lost and out-of-order messages will be received, although with generally low probability.

When they do occur, however, they happen in bursts.

Two pieces of information are carried in each speech data message which aid in the detection and correction of out-of-order or missing messages. These are the time stamp and the message sequence number. Both of these quantities should be monotonic increasing from message to message when the messages are properly ordered. The sequence number increases by 1 between messages. The time stamp increases by the time in milliseconds represented by the content of the previous message, which is a multiple of the block time of 19.2 msecs for non-silence messages and a variable time up to the nominal TIME-SEND-SILENCE for silence messages. Each quantity has a very finite limit in its range, and must be interpreted modulo that limit. In particular, the message sequence number can assume only sixteen values. Its use in ordering messages is thus limited if out-of-order or missing messages occur in bursts. The time stamp is limited to 32,768 values, completing a full cycle in a little over 32 seconds. If the typical non-silence message represents 135 to 154 milliseconds of speech, about 200 messages would be sent before the time stamp wraps around. Because of this increased resolution, and since it can provide the information needed to "ride-over" lost messages, we have chosen to rely on the time stamp for proper ordering of incoming data messages.

Since input of the digitized speech data to the processor may be buffered in two or more blocked buffers before it is received, the normal asynchronous timer cannot be used reliably to provide the time stamp. Instead, the known sampling rate of the analog-digital converter is used to calculate the time represented by each input block and the block index times this value provides the time stamp for a given block. The time stamp for a message is then the time stamp for the first block in the message, if the message is non-silence.

Since the NVP requires the time stamp to be expressed in milliseconds, if the block index overflows, a discontinuity in the time will result. This can be avoided by testing occasionally for the block index exceeding a selected integer such that the block time multiplied by the integer is a multiple of 32768. For example, with a block time of 19.2 milliseconds, any multiple of 5120 can be used. Whenever the selected value is exceeded, it can be subtracted from the index to avoid overflow.

For silence messages, the time stamp is chosen to satisfy two criteria. First, the initial silence message should have a time stamp derived from the first block whose parameters are not transmitted. This allows proper timing of the processing of this message by the receiver. Second, the time stamp of the last silence message should be strictly less than that of the first following non-silence message, to assist in proper ordering of these messages by the recevier. This requirement could create a problem, since the time stamp of the first non-silence message precedes the time at which the end of long silence was detected. To avoid this problem, a queue of parcels is accumulated whenever long silence is declared, this queue is limited to the number of parcels needed to provide the lead in to non-silence by discarding the oldest parcel whenver the queue limit is exceeded. The time associated with the oldest parcel in the queue at the time when a silence message is to be sent is used for the time stamp of the message. The oldest parcel is than discarded to avoid the equal time stamp problem.

As noted earlier, when the message currently being processed is not a silence message, the time stamp of the next message can be predicted from the time stamp of the current message by adding 19.2 for each parcel of parameters in that message. If the queue of input messages is maintained in order by time stamp, only the first message on the queue needs to be examined when a new message is to be processed. The following tests can be performed to detect out-of-order or lost messages:

1. If the time stamp of the new message precedes that of the message last processed, the message is assumed out of order and discarded.

2. If the time stamp of the new message is not greater than the predicted time stamp by more than a selected tolerance τ, the message can be processed immediately.

If the time stamp of the new message exceeds the predicted time by more than τ, the message is not processed. Since no new messages are being processed, the supply of synthesized data may be exhausted. In this case, the predicted time stamp should be updated to reflect "fill" time during which no new data is available for output. The input message queue should continue to be examined to detect either the arrival of the desired message or the passage of the time represented by a missing message.

The effect of this procedure is to maintain a "pseudo timer", the predicted time stamp of the next message, which represents the progress of the synthesizer through the incoming parcels. As long as the correct message has arrived before the last block synthesized from the previous message has been output, the pseudo-timer will keep up with real-time. When a message fails to arrive by the time the pseudo-timer matches its successor's time stamp, the missing message will be considered lost, but the pseudo-timer will stay in step. Only if a message is tardy, but arrives either ahead of its successor or before the time represented by its data has past, will the pseudo-timer slip and run slower than real-time.

Silence messages require some additions to the methods described here, since the time stamp of the succeeding message cannot, in general, be predicted from the time stamp of the silence message. The pseudo-timer could be maintained, using the time stamp of the message updated by the amount of null output generated for the silence. The next message would then be processed when its time stamp was not greater than the pseudo-timer value + τ. However, this would not take advantage of the silence periods to catch up with the real time input messages. Instead, silence messages can be processed immediately as they are received until a non-silence message is encountered. At this point, the TIME-AFTER-SILENCE delay is begun. Only when this time has past should the processing of the non-silence begin. The delay permits a short queue of incoming messages to develop, allowing some time for a possible out-of-order message to be received and re-ordered.

1.3.3 SUMMARY

Since the receiver's ability to correctly sequence incoming messages is dependent on a complete understanding of the transmitter's algorithm for preparing messages, we have discussed the assignment of time stamps as well as their use in message ordering. In this summary, we have included block diagrams and descriptions for both transmitter and reciever's message processing.

Transmitter

Let PTIME be a counter which is incremented by one each time a parcel of parameters is prepared for transmission. Each count of PTIME represents 19.2 milliseconds.

Let MTIME be set to the value of PTIME whenever a message is prepared for transmission. MTIME multiplied by 19.2 will be the time stamp for the next message.

Let SCNT be a counter of the number of consecutive parcels whose gain is below the silence threshold, SLNGTH.

During long silence, parcels continue to be collected until the time represented by the parcels is equal to the desired leadin time. If sufficient parcels are in the queue, the oldest parcel is discarded and MTIME is incremented each time a new parcel is produced.

When a silence message is sent, its time stamp is calculated from MTIME. MTIME is then incremented by one and the oldest parcel is discarded, regardless of the number of parcels on hand.

When long silence ends, the collected parcels form part of the first non-silence message. MTIME is used to compute the time stamp as before and then reset to PTIME. This assumes that the amount of leadin time is less than the maximum non-silence message time.

Receiver

Let STIME be the time stamp of the message currently being processed.

Let CTIME by the computed time derived from STIME by adding 19.2 milliseconds for each block of 128 points synthesized for output and for each block of null output generated during silence or while waiting for delayed messages.

Whenever a new message is needed for synthesis, the following steps take place:

1. The queue of input messages is ordered by time stamp, if not already in order. If no messages are present, exit.

2. The time stamp of the first message is compared with STIME, if it is less, the message is discarded and we begin at step 1 again.

3. If input silence has already been estalbished, and the new message is silence, set STIME=CTIME=time stamp, discard the message and exit.

4. If the new message is the first non-silence message, set CTIME to the time stamp of this message less the desired time after silence delay.

5. If the time stamp of the message is >CTIME+τ , exit.

6. Set STIME=CTIME=time stamp of the message. If the new message is silence, establish input silence and discard the message. Otherwise, unpack the parcels for processing by the synthesizer, discard the message, exit.

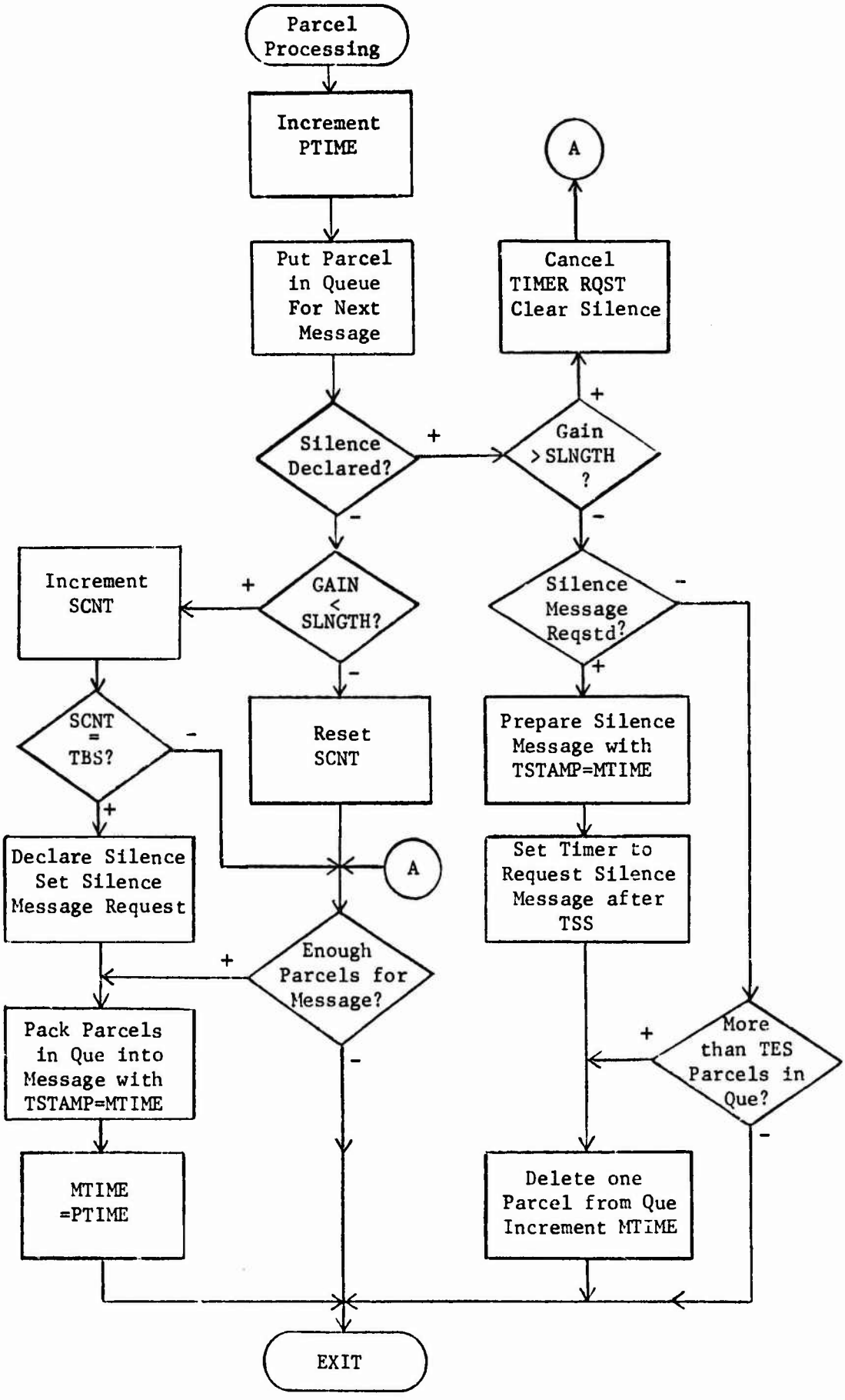


Figure 1-8. Block Diagram of Transmitter Message Preparation

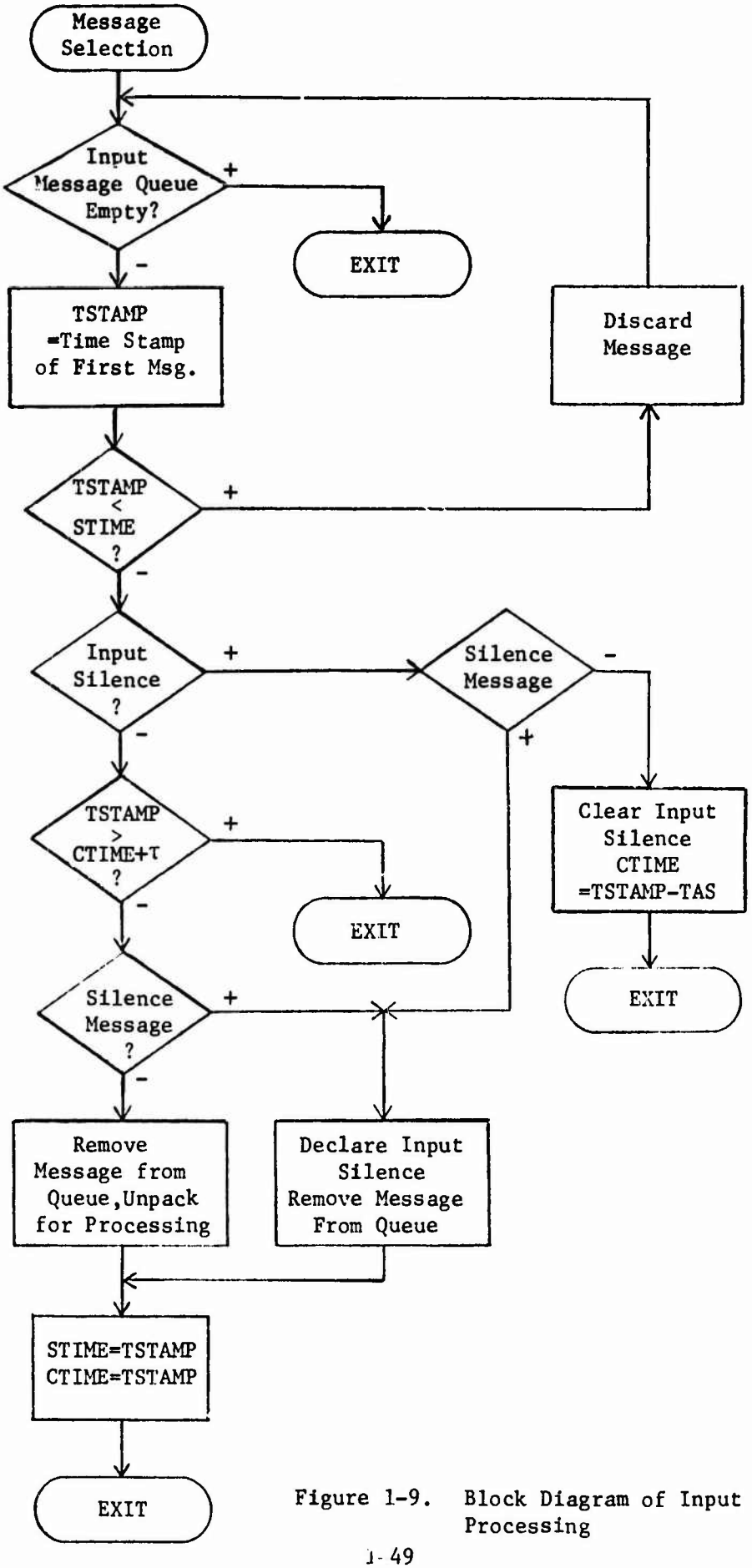


Figure 1-9. Block Diagram of Input Message Processing

1.3.4 APPENDIX

Network Measurements

The data reported here is from one experiment between CHI and Lincoln Labs on May 5 at about 12:15 PM PDT. The conversation lasted about ten minutes. Uncontrolled network messages were used by both sites for speech data transmission.

During this experiment, we recorded for each message received the time it arrived, the time it was removed from the input queue for processing, both measured by our system timer, and the message length, IMP-HOST leader and message header. In addition, the entry was flagged if the message was discarded because it arrived out of order and late. This data has been reformatted and a sample of one interesting portion of the record is shown in Table 1-3. The columns have the following meaning:

ORDER	0 If message was used, the TSTAMP of the message last processed if the message was discarded.
S	Blank for non-silence messages - for silence messages
COUNT	The four-bit count from the IMP-HOST leader
LEN	The number of parcels in the message, 1 for silence messages.
TSTAMP	The 15-bit time stamp from the message header.
RCVD	The time the message was received.
PRCSD	The time the message was removed from the input queue for processing.

All times are in milliseconds, modulo32768. Note that five messages were delayed about fifteen seconds, and some messages with later time stamps got through ahead of time. Also, seven messages (13, 14, 15, 0, 2, 3, 4, 7, 8) are missing.

Several graphs have been prepared from the data. Figure 1-10 shows the variation in message transit time from Lincoln to CHI. The pronounced drift during the course of the experiment was apparently due to a difference in clock rates for the transmitter and receiver. The local variation is generally then 500 milliseconds. No absolute time scale is intended for this graph.

Figures 1-11 and 1-12 plot the time each message spent in the input queue. Since it is desired that this queue provide sufficient buffering to smooth out the variations in network transit time, the cases where the message spent no time in the queue are particularly important. To clarify these points, the time in queue for silence messages was forced to 625 milliseconds in Figure 1-12, since silence messages are treated specially. The remaining cases of 0 queue time represent times when the synthesizer caught up with the input, and may have had to output fill blocks while waiting for a message. Obviously, there is a discrete change in these graphs. during the experiment, the time after silence delay was decreased from 500 milliseconds to 250 milliseconds. It appears that 250 milliseconds were not adequate to build up the necessary queue.

The occurrences of out-of-order or missing messages are shown in Figure 1-13. The abscissa in this case is the time stamp of the message, rather than the message number. A total of 20 messages was lost out of 1800 in this experiment. In addition, nine messages were received out of order; five of these have already been mentioned, the other four are identified on the graph. All out-of-order messages arrived sufficiently late that they were discarded.

The effective bandwidth of the system can be calculated as follows:

Number of silence messages:	264
Length of silence message in bits:	80
Number of bits needed for silence messages:	21120
Number of non-silence messages:	1536
Total length of non-silence messages:	54158 bytes
Number of bits needed for non-silence messages:	866528
Total bits transmitted:	887648
Total elapsed time:	620
Average bandwidth:	1432 bits/second

The time in silence was 411 seconds:

The time represented by non-silence messages was 209 seconds.

The total bits transmitted counts only those received by the HOST. It includes HOST-IMP and IMP-HOST padding totalling 16 bits/message. All measurements represent only the Lincoln-to-CHI half of the conversation.

Table 1-3. Sample Trace Record -- Messages 250-299

RECEIVE TRACE RECORD

(FCN IS THE FIRST MESSAGE

#LOOP	S	LEN	COUNT	TSTAMP	RCVD	PROCD
0		8	5	15939	15372	16069
0		7	6	16093	15616	16232
0		7	7	16227	15644	16356
0		7	8	16361	15732	16492
0		7	9	16496	16000	16628
0		7	10	16630	16184	16772
0		7	11	16765	16216	16912
0		7	12	16899	16292	17032
0		7	13	17033	16528	17176
0		7	14	17168	16552	17316
0		7	15	17302	16712	17460
0		7	0	17437	16784	17588
0		7	1	17571	16928	17712
0		7	2	17705	17180	17856
0		7	3	17840	17272	17980
0		7	4	17974	17352	18128
0		3	5	18109	17420	18252
0		1	6	18156	17432	18428
0		12	7	18531	18116	18616
0		7	8	18761	18144	18732
0		7	9	18896	18204	18856
0		7	10	19030	18592	19000
0		7	11	19165	18616	19124
0		7	12	19299	18704	19260
0		7	13	19433	19024	19396
0		7	14	19568	19056	19532
0		7	15	19702	19196	19676
0		7	0	19837	19424	19804
0		7	1	19971	19396	19936
0		7	2	20105	19492	20076
0		7	3	20240	19576	20220
0		7	4	20374	19848	20344
0		7	5	20509	20048	20480
0		7	6	20643	20076	20616
0		7	7	20777	20252	20752
0	-	1	1	25827	25196	25200
0	-	1	5	1124	324	376
0	-	1	6	3140	2416	2416
0		7	8	20912	3428	3432
0		7	9	21046	3468	3468
0		7	12	21449	3488	3488
0		7	10	21181	3588	3600
0		7	11	21315	3612	3612
0		12	9	8785	8164	8664
0			10	9015	8180	8768
0				9132	8292	8328
0			14	9264	8392	9040
0			15	9396	8648	9180
0			13	9528	8728	9328
0			11	9637	8916	9466

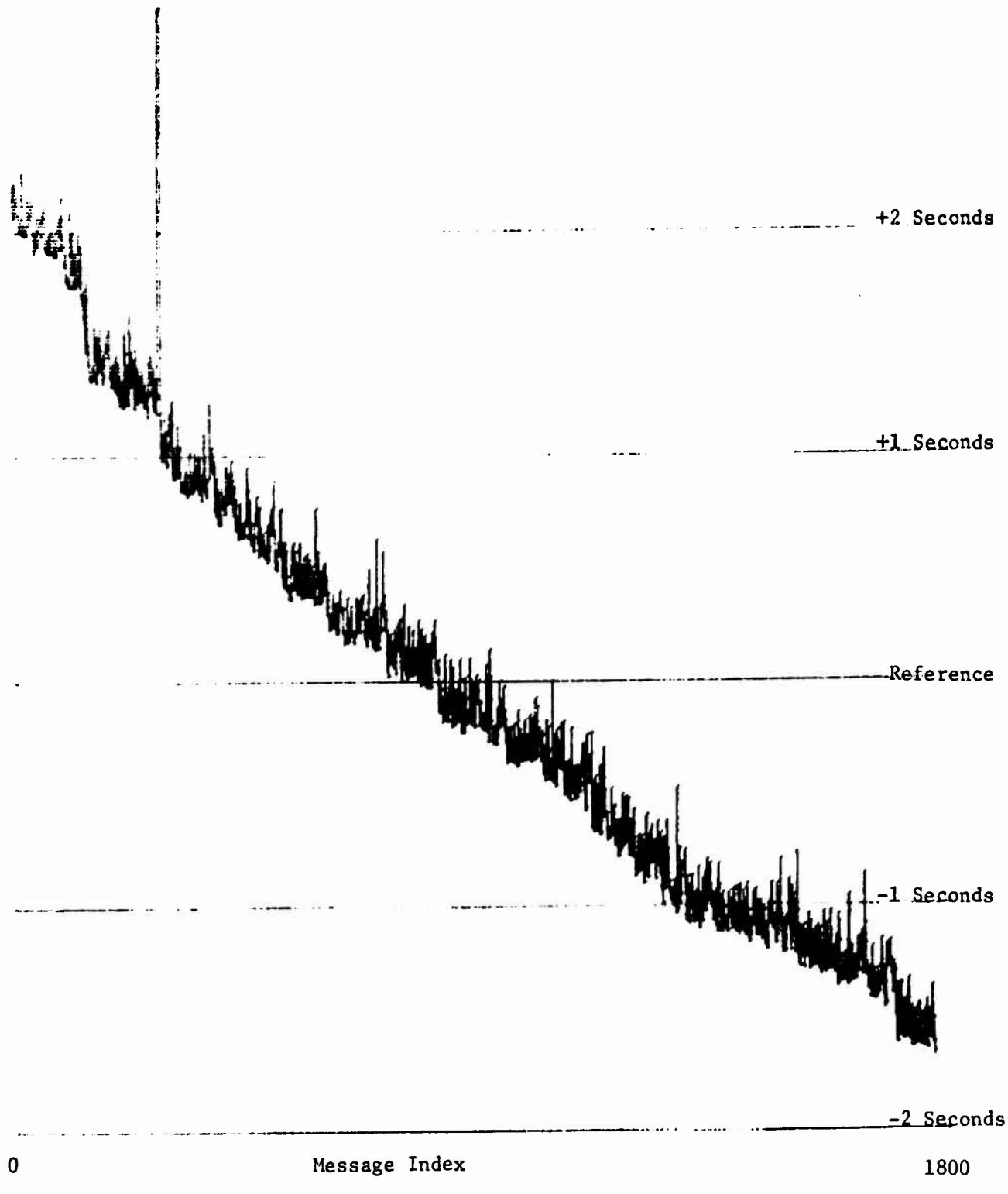


Figure 1-10.Variation in Time Received - Time Stamp, corrected for message length

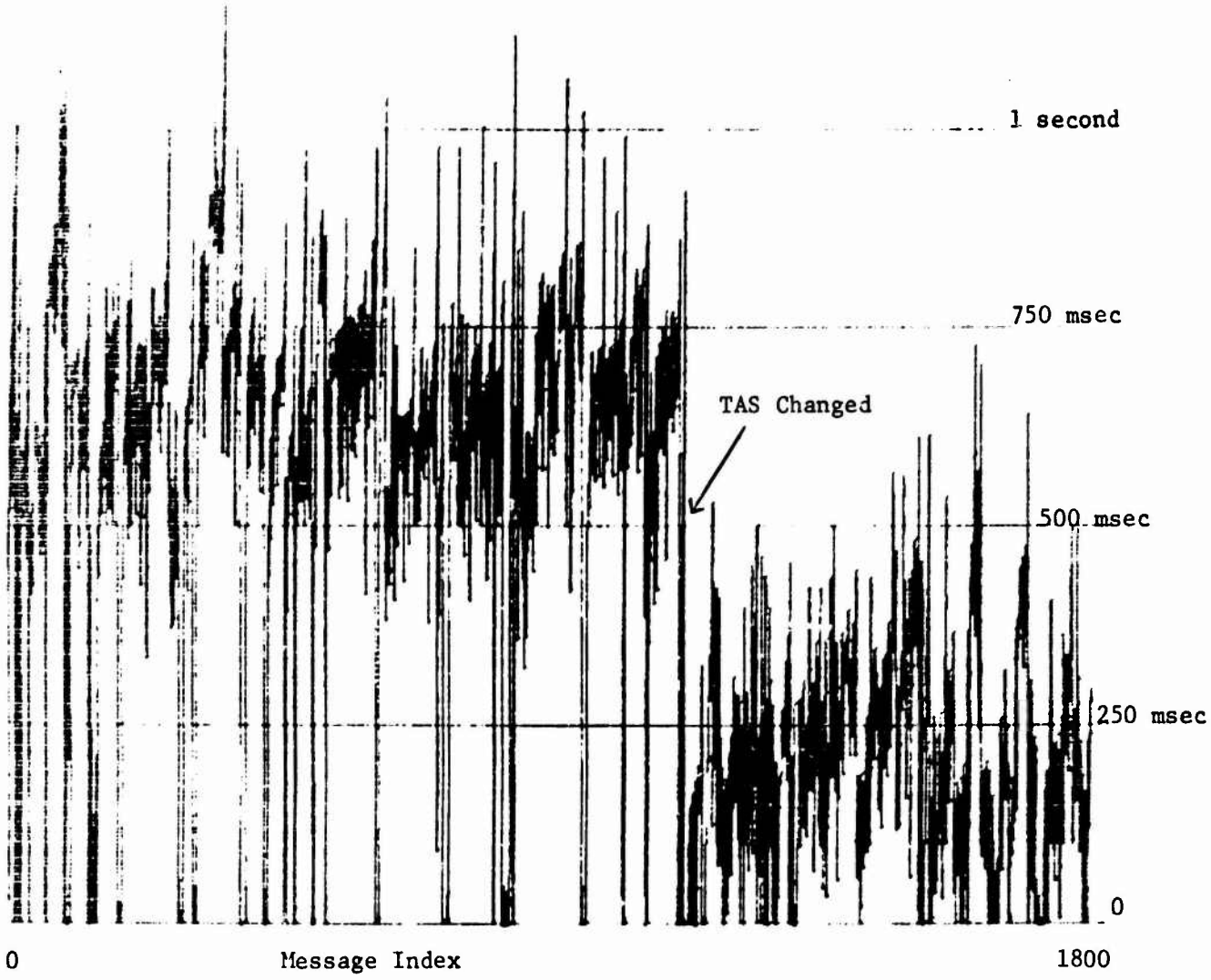


Figure 1-11. Time on Input Queue for Each Message

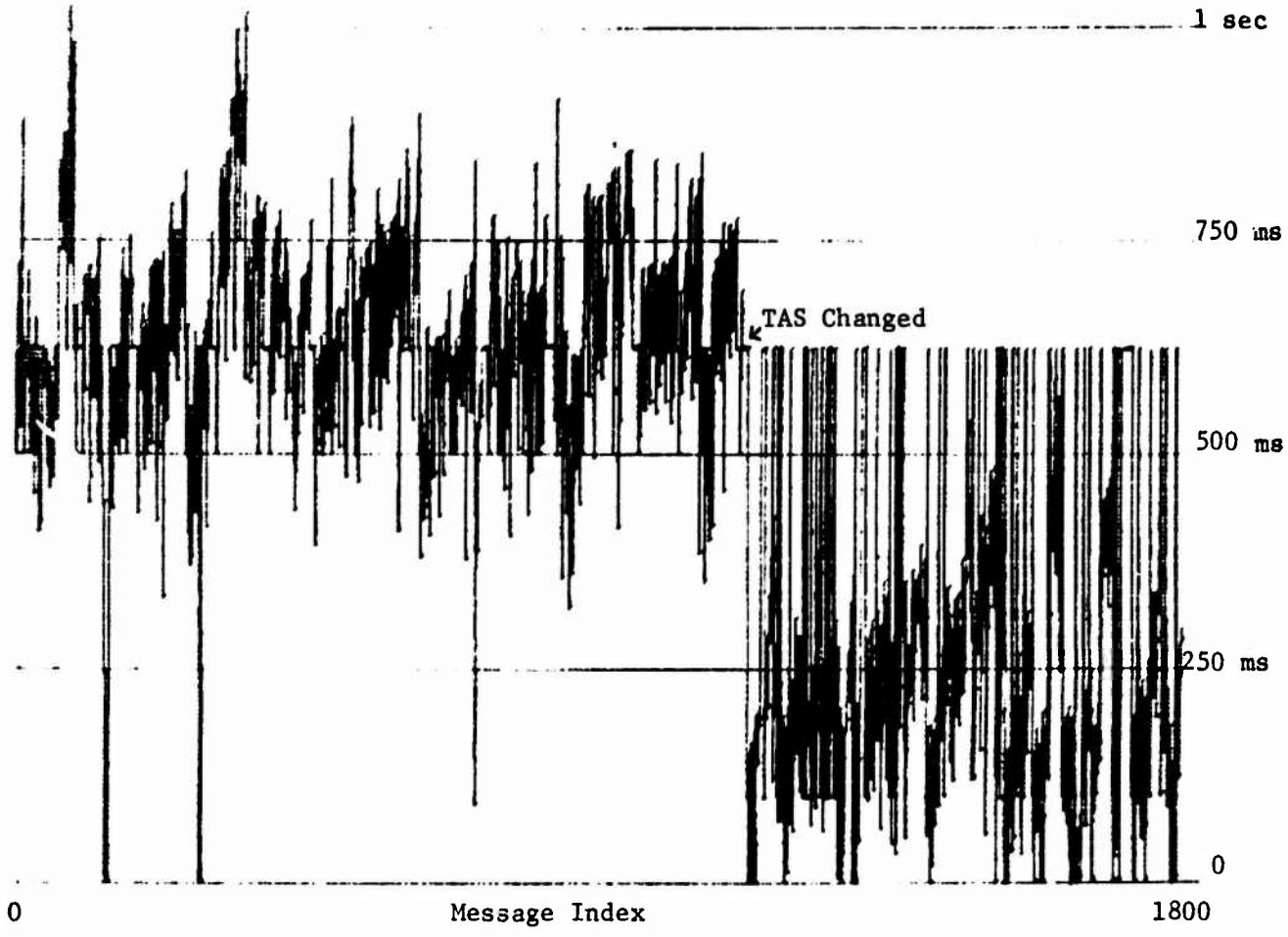


Figure 1-12. Time on Input Queue for Non-Silence Messages

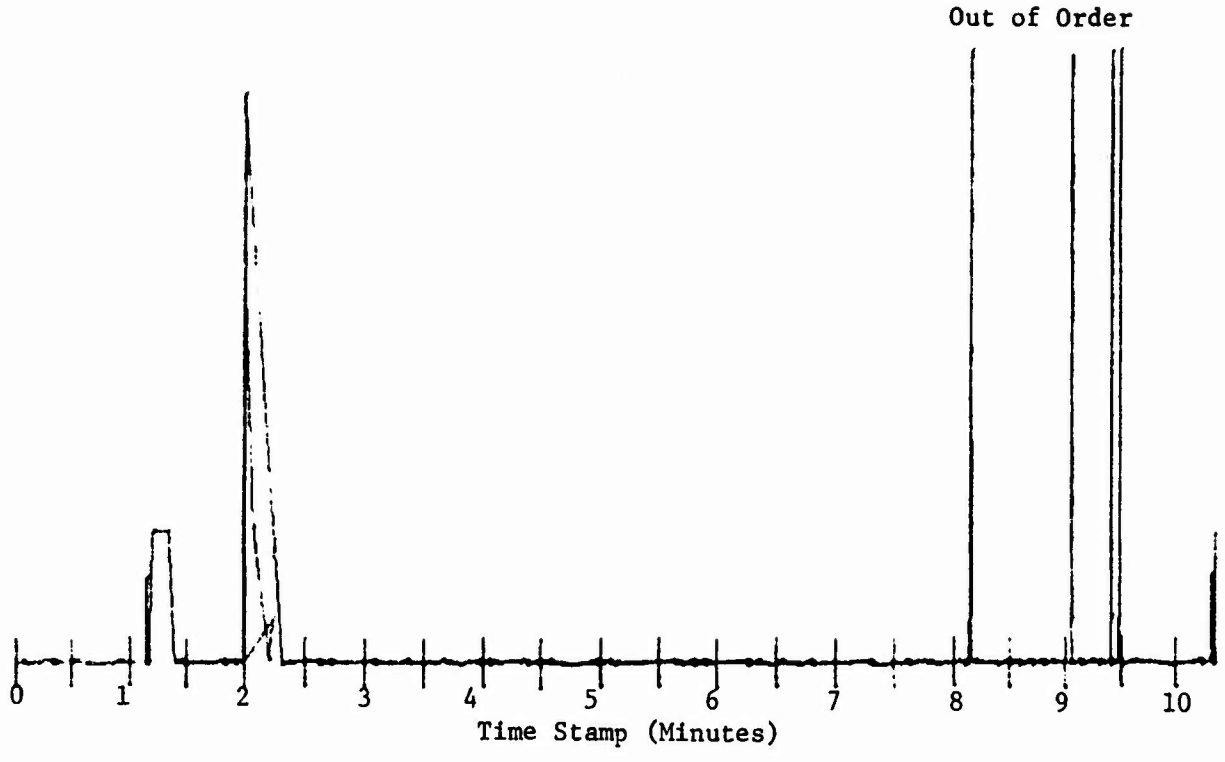


Figure 1-13. Lost or Out of Order Messages

1.4 RESULTS

The quality of our implementation of the linear predictive coding algorithm for speech data compression may be judged by listening to the tape recording enclosed as part of this report. All information recorded on the tape with the exception of segment three is synthetic speech, generated by our synthesis program and output through a 3.2KHz low pass filter.

The order of data on the tape is:

Segment 1: An introduction to the rest of the tape describing the equipment being used.

Segment 2: A short demonstration of real time LPC speech, including analysis, quantization and synthesis. A variety of speakers are used.

Segment 3: A digitally recorded copy of a tape of two British speakers. This recording has been low-passed through a 3.2KHz filter and sampled at 150 microsecond intervals using a 12-bit analog-to-digital converter. The data is then converted back to an analog signal through a 12-bit digital-to-analog converter and low passed through a 3.2KHz filter. This particular tape demonstrates a wide range in pitch and amplitude. In some sections the female voice exceeds the upper bound of the pitch detection algorithm used.

Segment 4: This is the result of processing the original tape of two British speakers by our original implementation of the LPC analysis and synthesis. This implementation does not include the parcel selection and K_1 criterion for voicing described in Section 1.1.1.2.

Segment 5: The result of processing the same tape with our present implementation of the LPC analysis and synthesis.

Segment 6: This is a sample of LPC speech compression on the ARPA network. This is an excerpt from an experiment with Lincoln Laboratory in December, 1974. Analysis was performed at Lincoln Lab using their TX2-FDP system. The parameters were encoded and transmitted to CHI over the network using standard (Type 0) network messages. The parameters were decoded and the speech synthesized and recorded from the analog

output at CHI using our system.

Segment 7: This sample of speech compression on the ARPA network was recorded in May 1975. Uncontrolled (Type 3) network messages were used for all data transmission. For this experiment, both sides of the conversation were synthesized by CHI. The speech from Lincoln Labs was analyzed at their site and transmitted to us over the ARPANET. The CHI side of the conversation was analyzed at CHI and then resynthesized for output as well as sent to Lincoln.

Prologue: A brief review of the method of preparing the tape concludes the recording.

REFERENCES

1. Markhoul, J. J. and Wolf, J. J., "Linear Prediction and the Spectral Analysis of Speech," Bolt, Beranek and Newman, Inc., Cambridge, Mass., Report 2304, August 1972.

2. Markel, J. D. and Gray, A. H., "A Linear Prediction Vocoder Simulation Based upon the Autocorrelation Method," IEEE Trans. on Acoustics, Speech and Signal Processing, ASSP-22, pp. 124-134, April 1974.

3. Markel, J. D. and Gray, A. H., "On Autocorrelation Equations as Applied to Speech Analysis," IEEE Trans. on Audio and Electroacoustics, AU-21, pp. 69-79, April 1973.

4. Markel, J. D., "Status Report to NSC Group on Linear Prediction Analysis/Synthesis Programs," NSC Note 20, May 15, 1974.

5. Markel, J. D., "The SIFT Algorithm for Fundamental Frequency Estimation," IEEE Trans. on Audio and Electroacoustics, AU-20, pp. 367-377, December 1972.

6. Markel and Gray, "Fixed Point Truncation Arithmetic Implementation of a Linear prediction Autocorrelation Vocoder," IEEE Trans. on Acoustics, Speech and Signal Processing, ASSP-22, pp. 273-281, August 1974.

7. Cole, E. Randolf, "Everything You Always Wanted to Know About Gain," NSC Note 45, October 11, 1974.

8. Markel, John D., "Two NSC Group Proposals and a Status Report," NSC Note 47, November 11, 1974.

9. Cohen, Danny, "Specifications for the Network Voice Protocol (NVP)," NSC Note 43, October 30, 1974.

CHAPTER 2. AUDIO SIGNAL PROCESSING HARDWARE

2.1 INTRODUCTION

An audio analog/digital signal processing system has been developed for use in speech research. The system acts as an interface between audio signal sources (tape recorders, microphones, speakers, etc.) and a computer, performing the necessary intermediate operations of amplification, anti-alias filtering, A/D and D/A conversion, data buffering and analog/digital ground isolation. A 20KHz bandwidth, wide dynamic range and simultaneous multi-channel I/O make the system useful in a more general context than that of speech data handling. Other potential applications are seen in the fields of high quality digital recording, in situ testing and calibration of acoustical transducers and transducer array data processing.

For interim use, a two-channel system was constructed with one input and one output channel. This is the system that was used for the CHI/LINCOLN LABORATORY network speech compression demonstration in December, 1974. It is described in detail in Section 2.2. In Section 2.3 the multi-channel laboratory system is described.

2.2 TWO-CHANNEL AUDIO SIGNAL SYSTEM

Figure 2-1 shows a block diagram of the audio system. The input channel centers around the amplifier/filter module, which is described in detail further on. Data can be taken from microphone or tape recorder. The standard microphone for speech input is the Sony ECM-280 Electret Condenser. For acoustical cavity data a Telectret 5336C* microphone is used. The tape recorder can be used as a source of pre-recorded analog data, or to record in-lab or network speech experiments. The sample/hold and A/D units are discussed in Section 2.3.

The output channel is used for playback and/or recording of digitally stored data. The D/A converter drives one of the output filters which feeds into a line level input of the Kenwood amplifier. The filters are identical to those in the input channel. The Kenwood provides tone, volume and function control for the output channel.

Figure 2-2 shows a block diagram of the amplifier/filter module. The microphone input stage is 1/2 of an LM381 low noise IC, with gain selectable by a passive divider on the preamp output. A 4-watt power chip (LM378) provides a fixed voltage gain of 20db and current to drive the filters. The overall amplifier voltage gain is selectable at 66, 63 and 54db. The unfiltered output is flat (±3db) from 10Hz to 20KHz. Equivalent input noise voltage is 3μV for the 20KHz band. A battery power supply eliminates ripple in the input stage. The amplifier is not optimized for use of battery power (quiescent current drain ≈20ma) but this design compromise is not necessary in the newer system, where more care is taken with line power supplies (see Section 1.3) in order to eliminate the ripple problem.

The anti-aliasing filters are selectable with -3db frequencies of 3.2KHz, 4.9KHz and 10KHz; the output is down -66db/octave for these filters. They have a (compensated) shunt impedance of 333Ω.

---

*These are very high quality miniature microphones. The sensitivity is 6.25 times greater than the Sony and the S/N is 3db greater. The disadvantages of the 5336C are that it requires power (500μA at 9V) and that it is not available with cable attached. Some care must be taken in cabling to avoid 60Hz pickup.

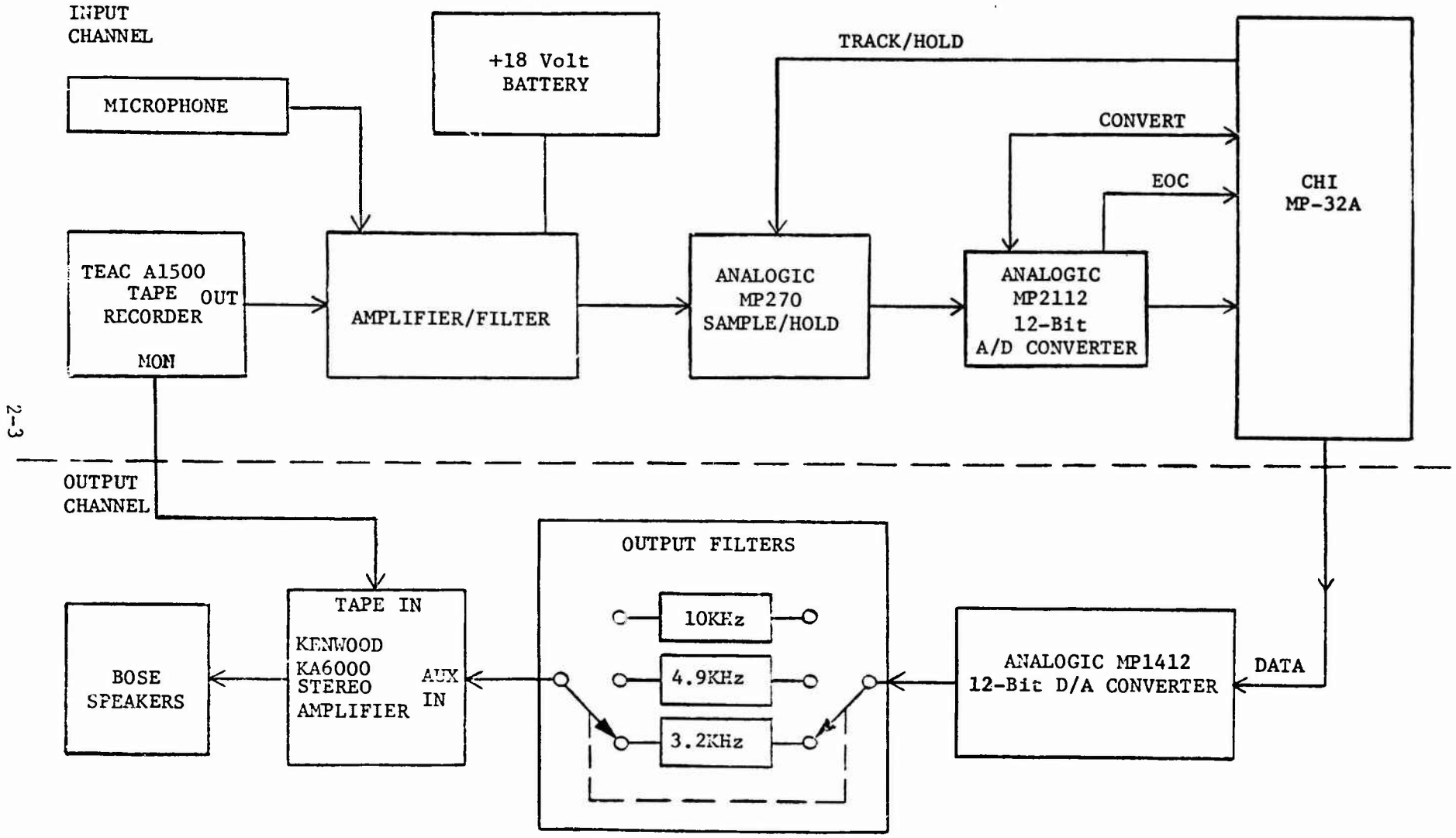


Figure 2-1. Block Diagram of Two-Channel Audio Signal System

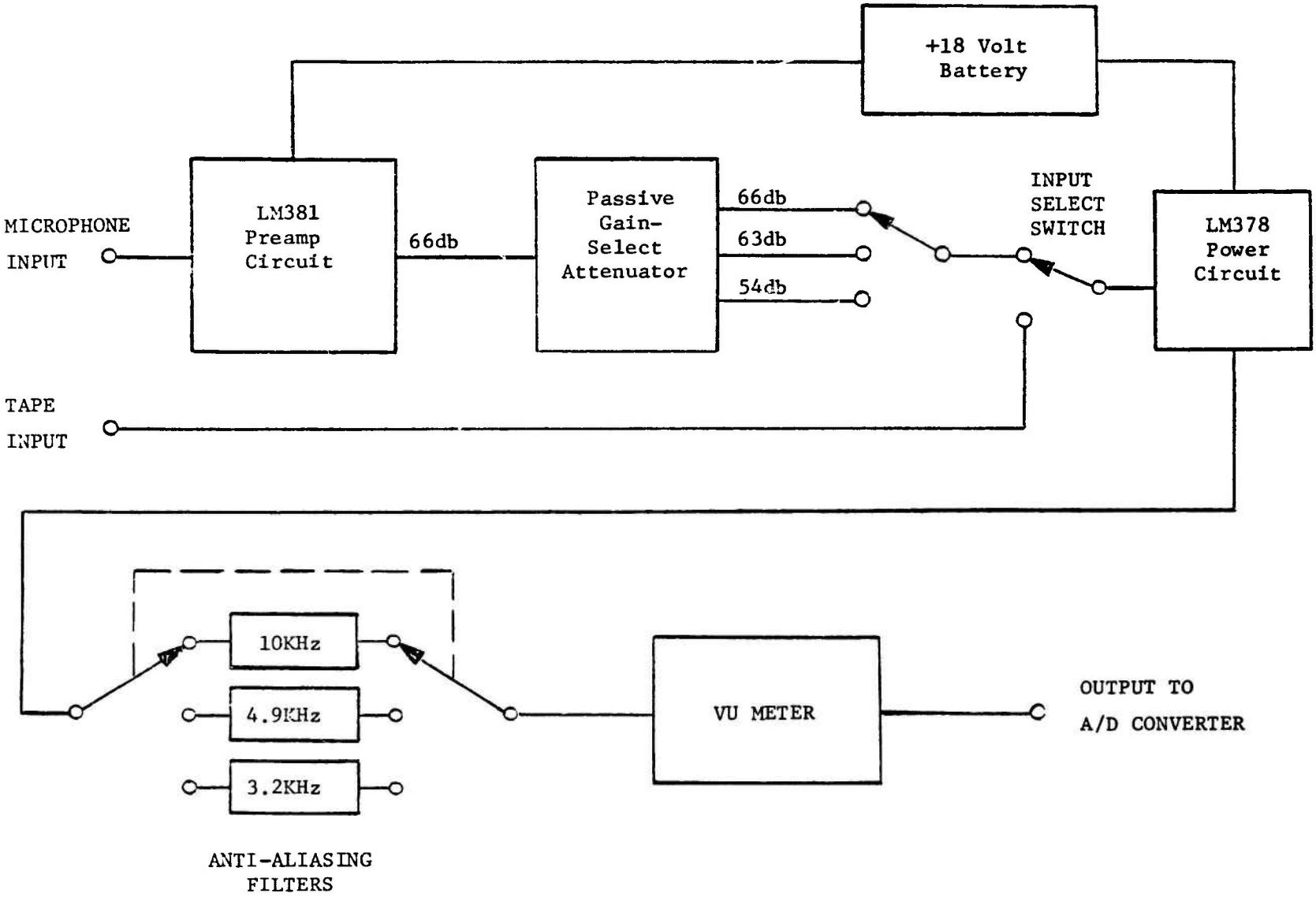


Figure 2-2. Block Diagram of Audio Amplifier/Filter

2.3 MULTI-CHANNEL AUDIO SIGNAL SYSTEM

2.3.1 GENERAL DESIGN CONSIDERATIONS

The immediate goal of this project is a high quality A/D and D/A conversion system for handling speech data and acoustic laboratory data. In both cases the source transducer is typically a microphone, the output transducer is a speaker. At times a tape recorder may replace the source and/or output transducer.

The requirement for speech recording and playback is one A/D channel with amplification and one D/A channel capable of driving a speaker. The acoustic laboratory requirement is a multi-channel A/D input for handling data from an array of microphones and a single D/A channel for output. The initial need for several identical channels coupled with an awareness of the muteable nature of hardware needs for experimentation, resulted in a modular system structure. Each channel is a self-contained chassis which slides into a rack mount frame (see Figure 2-3). This organization provides a flexibility for addition to and modification of the initial system. For example, a programmatically controlled anti-aliasing filter can be added later as a separate module to augment the fixed band filters. Similarly, a stepper motor control for positioning transducers in the laboratory can be added as a single extra module.

In designing a digital recording system one impacts all of the problems associated with quality analog design and, in addition, must deal with undesirable artifacts of the digitization process. (A list of parameters relevant to specifying such a system is given in Appendix 1.) One such problem is the contamination of low level analog signals by digital noise. This noise, generally, is at the clock frequency, but can also correlate with slower periodic operations, such as memory↔disk data transfer in a looping program. This type of noise may appear as a chirp or glitch in an audio playback. One means of dealing with this general class of problems without having to treat each one specifically is to isolate analog and digital grounds and use separate power supplies. This approach is implemented in the present system by optical isolators. The specifics are discussed in the sections describing the different modules.

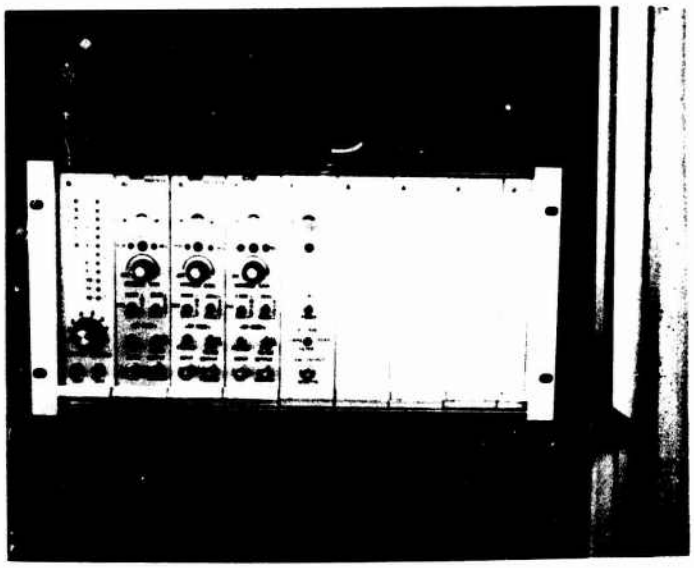
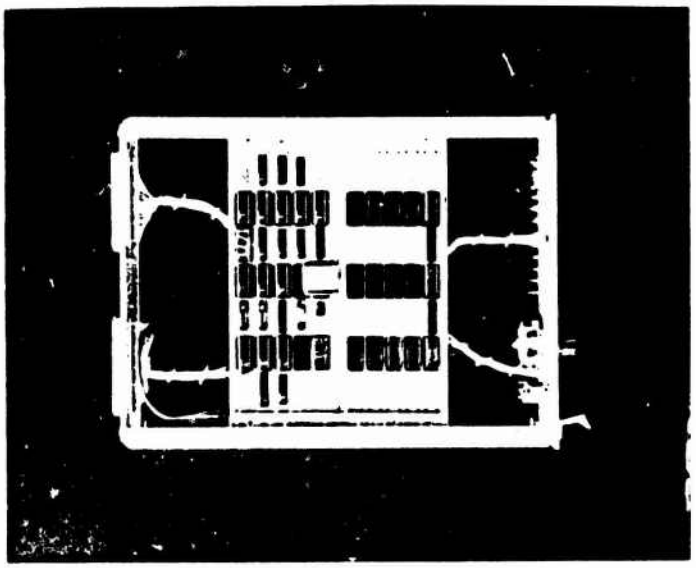


Figure 2-3 Photograph of Multi-Channel Audio Signal System



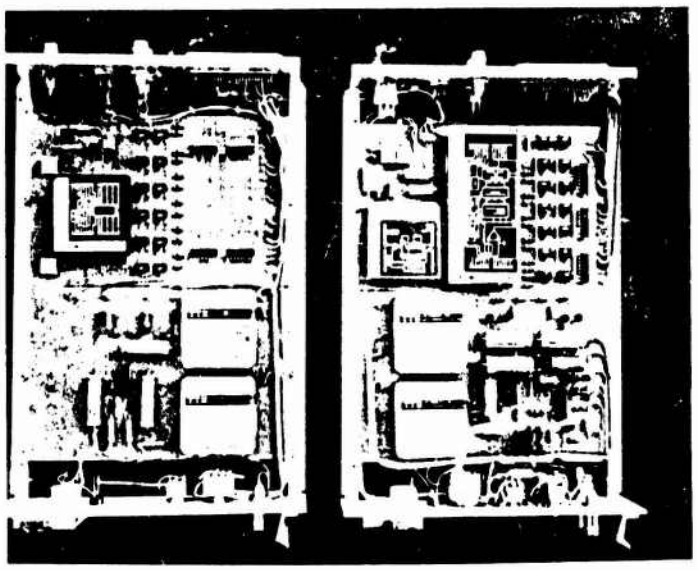


Figure 2-3a. Internal View of Modules; Control (top), D/A (left), A/D (right)

For control and interface logic, Tri-State [1] was selected rather than open collector for the following reasons:

i) active pull-up in Tri-State as opposed to passive pull up in open collector,

ii) one set of data lines from computer to laboratory can serve all instrument modules with Tri-State. Thus, for 16-bit resolution and N modules, Tri-State results in a saving of (N-1)x16 twisted pairs for data transmission.

Figure 2-4 shows a block diagram of the data acquisition system. The daisy-chain connection of modules is made possible by the Tri-State [1] logic. Modules in the rack mount frame (Figure 2-3) are shown enclosed by a broken line in Figure 2-4. As can be seen, up to eight modules are allowed in the present structure, although nothing prohibits the addition of another rack module. Three of the eight modules are undefined at present. The cable from the computer (CHI-MP-32A) to the data modules contains twisted pair which are assigned as follows:

16 pair: bi-directional data lines
6 pair: address lines
1 pair: interrupt

The cable from the control module to the I/O modules contains:

16 pair: bi-directional data lines
6 pair: address lines
1 pair: load command
1 pair: convert command
1 pair: enable command

1.3.2 TIMING

The easiest way to understand the typical operating mode of the system is to look at the timing diagram. But first a digression must be made to discuss bandwidth requirements, since this is a principal factor in setting timing constraints.

For network speech compression work a bandwidth of 3.2KHz is used, but for purposes of experimentation, a wider band system is desirable. Fricatives may contain information up to 10KHz, so this is a reasonable specification for high quality speech reproduction. For acoustic laboratory work, a band of 20KHz covers the entire audio range.

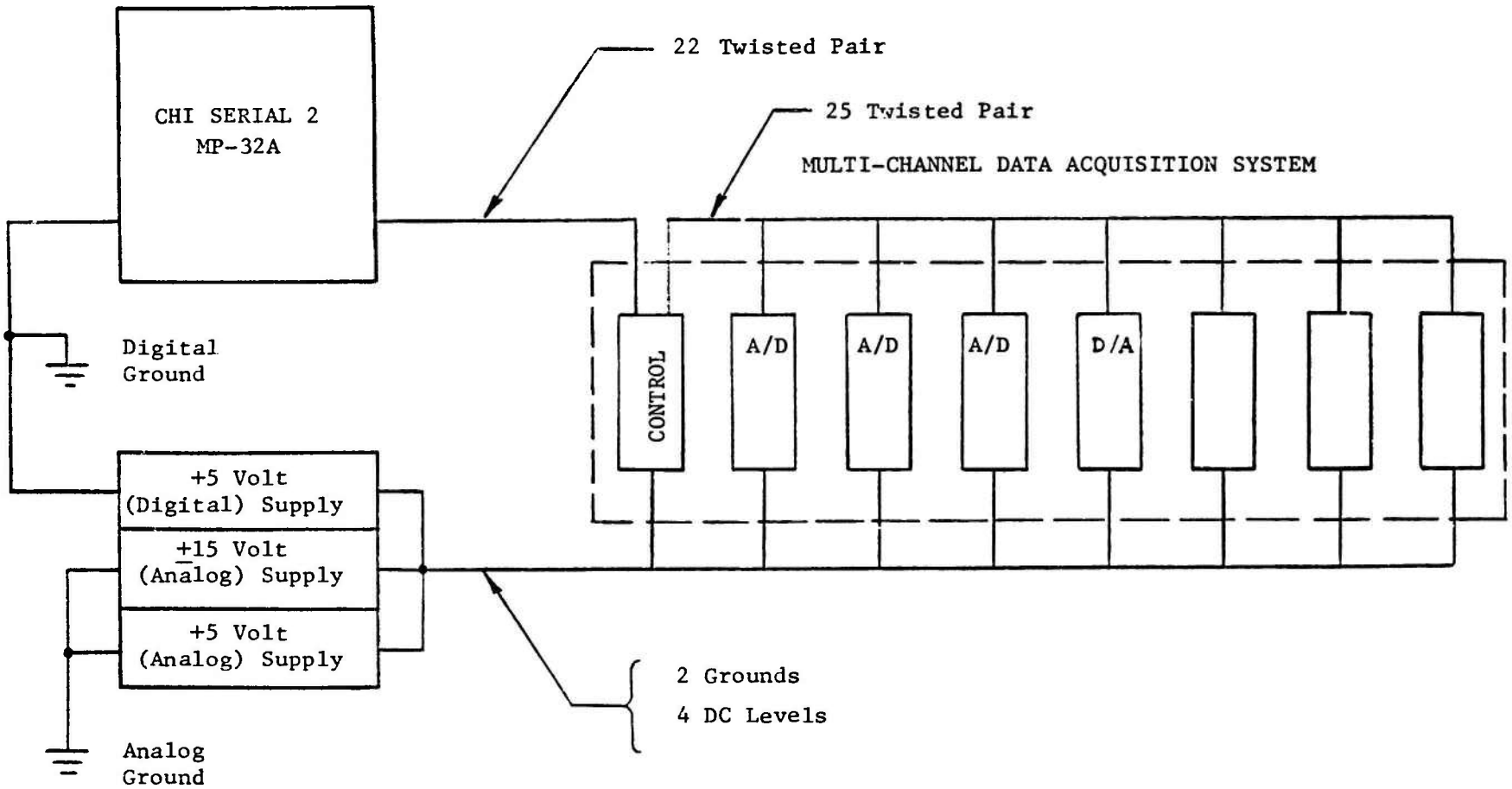


Figure 2-4. Block Diagram of Multi-Channel Audio System

A 20KHz band requires a sample time of 25μsec. However, for a packed spectrum, a margin is desirable to avoid foldover near the Nyquist frequency. For this reason, the minimum sample period is set at 20μsec. This strictly corresponds to a bandwidth of 25KHz. To meet the laboratory requirements, the D/A output and A/D inputs must be processed every 20μsec. The ability of the computer to meet the memory requirements and service interrupts at this data rate is one potential problem. Another is the conversion and transfer rates within the analog system.

First consider the computer memory restrictions in the present system. The MP-32A and AP-90 have high-speed memories of 64K and 8K 16-bit words respectively. By clearing these and running a data input program from the AP macro pad, 74K words of storage are available for incoming data. For a sample time t_s and an N channel acquisition system it is then possible to collect data for a time T, where

$$T = \frac{74000}{N} \times t_s. \qquad 2.3.1$$

For example, at t_s = 20μsec and N = 2, one finds T = 740msec. This is an interesting number; a stereo recording with 25KHz Nyquist frequency will fill 74K words of memory in 740msec! For in situ acoustical testing this is a reasonable number. If one is not interested in room reverberations, but in the behavior of some transducer in the room, the data acquisition should stop after the first echo from the room boundaries. Since sound travels 1 ft./msec, this echo time is usually <50msec; and, therefore, the above calculated 740msec is not a severe restriction.

In the case of speech data with a 5KHz band and a single input channel, T = 7.4sec. Clearly, then, it is necessary to move data to disk storage during a recording. This, in fact, is done with very little problem at speech data sample rates. However, for higher sample rates or multi-channel operation it is necessary to be aware of data flow rate limitations. The MP-32A will interface up to eight Type 2314 disk drives. One such drive contains 200 cylinders of 20 tracks each with 3600 16-bit words and a header on each track; for a total of 14.4M words/disk. The disk rotation time is 25msec, so the average read/write time for a word

is 6.9μsec. This number would set the throughput rate, except for the fact that one revolution is missed in the read/write cycle at every cylinder boundary. With 20 tracks/cylinder, this means 25msec lost every 20 tracks. To continue data acquisition across this 25msec, a buffer in high-speed memory is necessary.

Taking into account that one of every 21 disk revolutions is lost, the minimum read/write time per word increases from 6.9μsec to 7.24μsec, This corresponds to a throughput rate of .138 words/μsec. The input rate is N/t_s, so to avoid overflow of the buffer,

$$\frac{N}{t_s} \leq \frac{1}{7.24} = .138 \frac{\text{words}}{\mu\text{secs}} \qquad 2.3.2$$

For 5KHz speech data from a single input channel, $N/t_s = 10^{-2}$ words/μsec, so the criterion is easily met. The time in which such a recording fills an entire disk pack is T = 24 minutes. At t_s = 25μsec, two input channels are allowed, so by writing on to disk an entire pack of data can be taken from two channels. The maximum record time for this case is T = 3 minutes.

The above discussion reveals the data flow restrictions imposed by transfer rates in the computer. In addition, the analog hardware imposes some limitations. The time for sample/hold settling and A/D conversion is ≤10μsec. All channels convert simultaneously and are then read out; therefore, all N channels must be read in a time (t_s-10)μsec. This constrains the number of channels allowable at a given sample frequency to be

$$N \leq \frac{(t_s-10)}{7.24} = .138(t_s-10) \qquad 2.3.3$$

where t_s has units of μsec in this expression. Notice that this is more restrictive than Eq. 2.3.2, in which A/D conversion time was ignored. Eq. 2.3.3 expresses the hardware limitations on N and t_s. For a given choice, the limit on total record time with one disk pack for storage is given by

$$T = \frac{14.4t_s}{N} \text{ sec.} \qquad 2.3.4$$

For a given data rate N/t_s, the memory buffer size can be specified. In order to save the data accumulated during one disk revolution, a buffer of N_B = 25msec x N/t_s is required. Now, one revolution is lost at every cylinder boundary and also any time the buffer becomes empty. The worst case occurs when the buffer becomes empty while the disk is writing the end of the last track of a cylinder. In this case, a 50msec buffer is necessary to handle two sequential lost revolutions of the disk. So the buffer length for a given data rate must be

$$N_B = 5x10^{-2} x \frac{N}{t_s}$$

Figure 2-5 is a timing diagram for the system. The operation is asynchronous to the computer with cycling determined after initiation by an internal clock. In the case illustrated, the sample time is 20μsec. During the first 2μsec the sample/hold amplifiers in the A/D channels are settling into the hold mode. The following 8μsec is the A/D convert time. At the end of this period an end-of-convert interrupt (SR05J*) is sent out and the A/D and D/A buffers are simultaneously loaded with new data. The computer then has t_s-10 sec to service all of the analog channels, where t_s = sample time in μsec. The details of the buffering are discussed in later sections which deal with the specific modules.

2.3.3 CONTROL MODULE

The control module performs several functions in the data processing system. It provides local asynchronous timing for the data handling modules, which includes supplying logical control for sample/hold amplifiers, data buffers and A/D and D/A converters. From the front panel an LED display indicates the current status of the address and data lines, (see Figure 2-3). A self-test mode is also included, which internally provides the synchronization normally supplied by the computer. This mode is activated by a single-switch on the front panel and allows debugging and calibration independent of the computer.

Figure 2-6 shows a block diagram of the control module. The 16MHz oscillator output is divided by four to provide a 250μsec clock for local timing (CK250). In the normal mode, a number corresponding to the sample

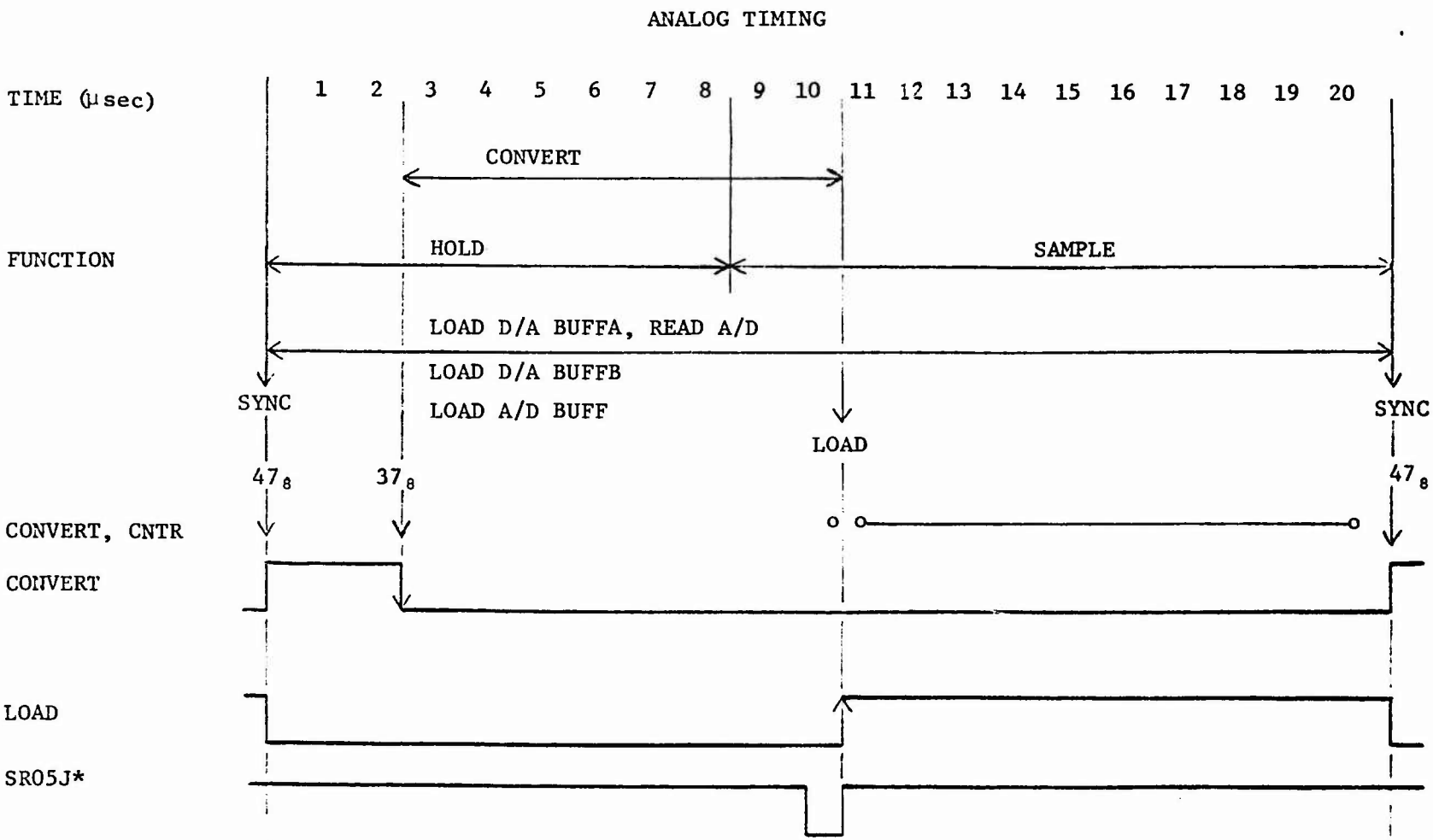


Figure 2-5. Timing Diagram for Multi-Channel Audio System

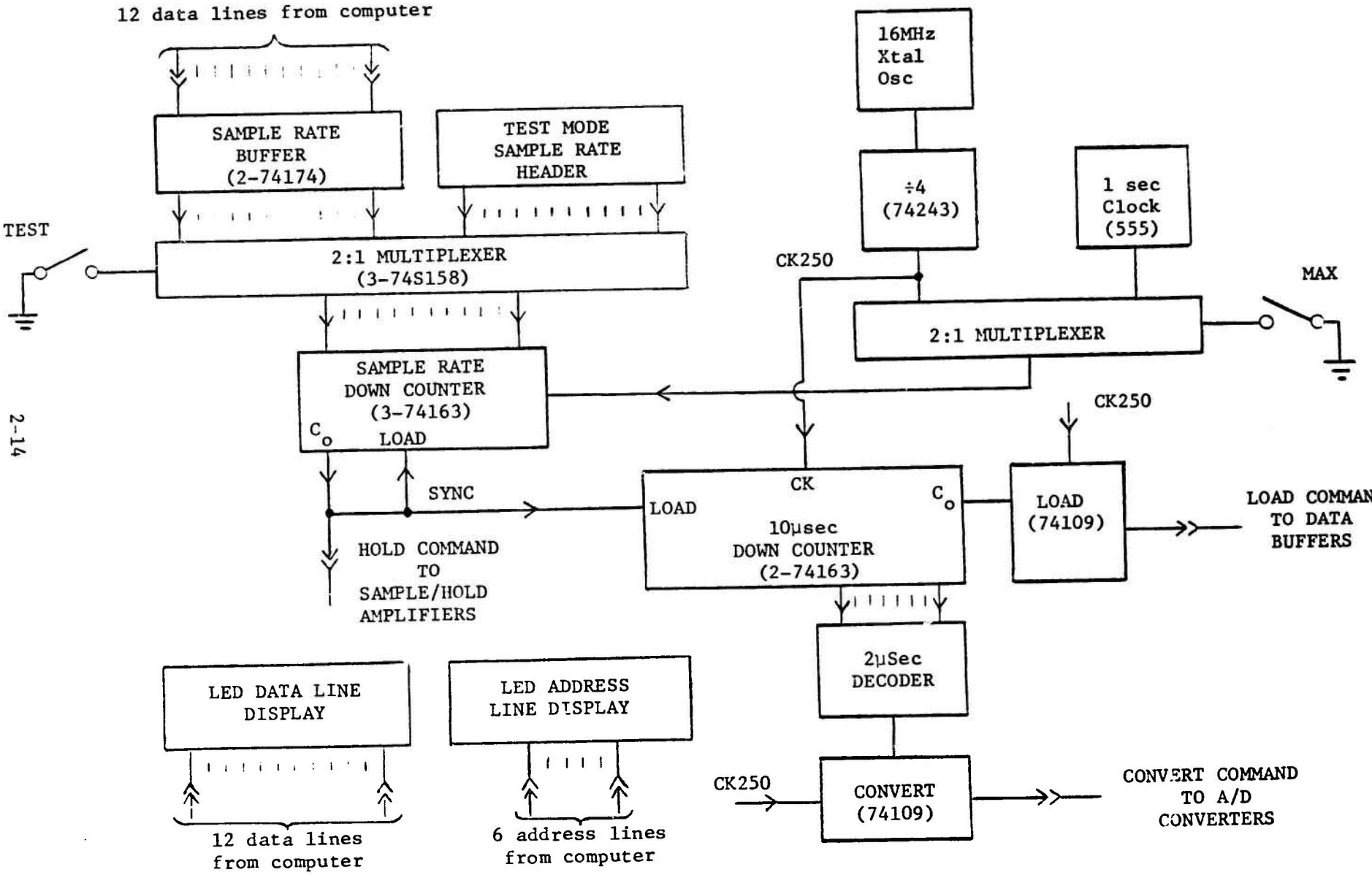


Figure 2-6. Block Diagram of Control Module

time is loaded into the sample rate buffer. This is accomplished by putting the number on the data bus and enabling the input buffer by specifying the address of the control module on the address lines. The sample time is restricted by the buffer length to be between 1 and 4096 clocks, or, 250nsec to 1msec.* A one-clock resolution is obtained within this range. The sample time is loaded into a down counter. When the counter output reaches zero the load line is dropped, putting the sample/hold units into the hold mode. At this time the 10μsec down counter is started. When this counter counts 2μsec the convert line is dropped, starting the A/D conversion. The 2μsec is a settling time for the sample/hold amplifiers. When the 10μsec counter reaches zero, the load line is raised which transfers all A/D outputs to their respective buffers, transfers all D/A buffer A contents to their respective buffer B and returns the sample/hold amplifiers to the sample mode. At this point an interrupt (SR05J* on Figure 2-5) is sent to the computer which then has (t_s-10)μsec to read all A/D buffers and load all D/A buffers

Closing of the TEST switch places the system in a self-test mode. In this mode the sample rate is determined locally by internal wiring of one header. The nominal setting is 20μsec, but any sample time within the range of the system (20μsec to 1msec) can be specified for use in this mode. The MAX switch slows the internal clock from 250nsec to 1 sec, the latter being supplied by a 555 timer. This allows visual monitoring of the address and data lines by the LED display, a useful option when testing, calibrating or modifying the system.

2.3.4 ANALOG INPUT MODULE

This module performs the following functions: amplification, anti-alias filtering, A/D conversion and data buffering. A block diagram is shown in Figure 2-7. The input signal is first amplified by a two-stage variable gain amplifier, then filtered by a pre-selected bandpass filter (TT Electronics, J83C). The amplified, filtered signal is fed into a sample/hold amplifier which drives the input of a 12-bit A/D converter. The 12-bit output is fed through optical isolators (Monsanto, MOC5000) into a Tri-state

*In practice, the lower limit on sample time is fixed, not by the clock resolution, but by sample/hold settling time and A/D convert time, which in this system amounts to about 10μsec.

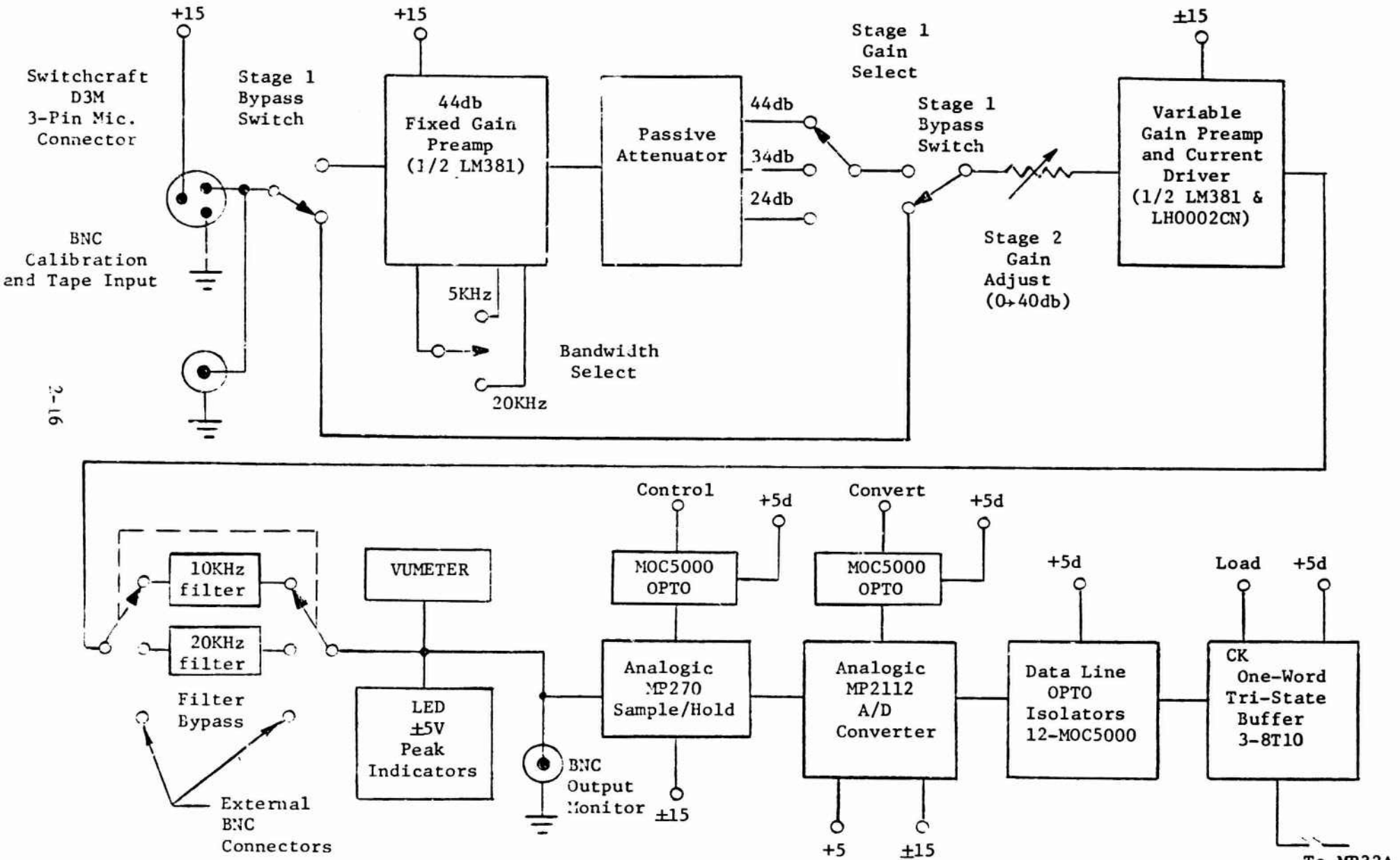


Figure 2-7. Block Diagram of Analog Input Module

buffer which holds the data until the computer services the data-ready interrupt. Notice that two different +5 volt supplies are required in this system. The supply marked "d" on Figure 6 supplies devices on the digital side of the optical isolators. This permits separation of analog and computer grounds.

The amplifier stage of this module is intended to perform optimally for microphone data which is characterized by about a 2mV amplitude and a 20KHz band. However, the wide dynamic range also permits input levels from tens of μvolts to volts. The external constraints which most strongly effect the specification of this amplifier are the source impedance, noise and output levels, the dynamic range of the sample/hold amplifier and the resolution of the A/D converter.

For purposes of this discussion, the term "amplifier noise" refers to the equivalent input noise voltage of the device, the assumption being that the source impedance is low enough that the equivalent input noise current does not contribute significantly to the overall noise. A reasonable goal for preamplifier device noise specification is arrived at by considering the source noise. The rule of thumb is that the equivalent input noise voltage, e_N, should be within $\sqrt{2}$ of the source noise voltage for optimization. Consider the Sony ECM280 microphone as a source. The inherent noise is ~30db SPL (re 200 μdynes/cm^2). The sensitivity is -55dbm which translates 160μV/dyne/cm^2. Combining these two facts, the noise voltage is found to be about 1μVolt. This is for the entire band of the microphone. The TEM 5336 microphone has a comparable noise level. It can also be noted that if a microphone is to be operating in an environment where the ambient noise causes a microphone signal in excess of the inherent microphone noise, then the ambient noise level can be used as a limit for design, rather than the inherent device noise. It appears that most microphones are designed with inherent noise within $\sqrt{2}$ of ambient levels in a quiet environment (20-30db SPL).

Designing an amplifier with input noise of 1 μVolt over a wide band is not possible with standard integrated circuits. However, one chip, the LM381, meets the requirement. This is a very high quality stereo preamplifier chip with <1μV input noise and about .1% harmonic distortion. It provides a gain of up to 60db with a 20KHz band. A very detailed discussion

of this device is given elsewhere [2].

The sample/hold amplifier has an input range of ±5 volts, so a typical amplifier gain should bring a microphone level up to ±5 volts. This means a gain of about 66db. In practice, a fixed gain amplifier is not a flexible module, so a wide range of gain is made available by separating the amplifier into two stages. It is well known to amplifier designers that the noise figure (NF) of a cascaded system is given by

$$NF = NF_1 + \frac{NF_2 - 1}{G_1} + \frac{NF_3 - 1}{G_1 G_2} + \ldots$$

where NF_1 is the noise figure of the first stage, NF_2 the noise figure of the second stage and G_1 the gain of the first stage. In practice this means that if NF_1 is low and G_1 large then NF_2 is not important. With this in mind, the first stage is a very low noise configuration, which is only possible with a fixed gain. The output of the first stage is divided and selected by a three-position switch. The second stage noise figure is not so important so this can be made a variable gain configuration. For line level inputs the first stage can be bypassed. The equivalent input noise voltage of this amplifier is measured to be about 3μV, which is fairly close to the original goal of 1 μVolt.

Another number to be considered is the quantization noise of the A/D conversion. Roughly speaking, if the analog noise at the A/D input is comparable to this, the system is optimum. The quantization noise, e_Q, has been derived by many authors and is given by the expression

$$e_Q = \frac{Q}{\sqrt{12}} \; ; \; Q = \frac{R}{(2^n - 1)}$$

where R is the output range of the A/D converter and n is the number of bits. For the converter in this system, R = 10 volts, n = 12, so e_Q = .7mV. Now, the amlifier noise is about 3μV. If the gain is set at 60db, the noise at the A/D input is 3mV, which is an acceptable number, although slightly high.

The modules following the amplifier in Figure 2-7 are fairly self-explanatory. However, one comment on the filters is useful. Notice that the bypass mode allows external substitution of an arbitrary filter. It also allows input from external amplifiers to bypass the input amplifier and be converted directly.

One closing comment concerning the use of an integrated circuit pre-amplifier in this design. Some purists claim the quality of these devices cannot compare to discrete amplifiers. The autnor feels that this is a reference to Transient Intermodulation distortion (see Appendix 1) and the nature of the overload distortion in integrated circuits. The TIM distortion has been observed on a scope but never audibly by the author in spite of fairly careful comparative subjective listening tests. It is admitted that FET devices ultimately have to be superior because of their square law transconductance curve which in theory only allows second harmonic distortion. Bipolar devices (including the LM381) have a diode (exponential) characteristic, which gives all harmonics as distortion products. These are very fine points, however, and probably have very little bearing on speech reproduction.

2.3.5 ANALOG-OUTPUT MODULE

A block diagram of the analog output module is shown in Figure 2.8. Digital data is loaded through a two-stage buffer and optical isolators into a D/A converter. The converter output supplies a unit gain driver capable of driving an 8Ω load to $\pm$5V.

The output filters smooth the staircase output of the D/A converter. As in the A/D module, an external filter may be utilized via the filter bypass terminals on the back panel. The 10KHz and 20KHz filters (TTE-J82C) are built into the module; the filter mode is selected by a three-position switch on the front panel.

The ground isolation problem is treated as it was in the A/D module. Twelve optical isolators, one for each digital data line, are inserted between the digital buffers and the A/D converters. Two 5V power supplies are necessary, one tied to digital ground and one to analog ground.

The power drive stage is 1/2 of an LM378. This is a stereo power amplifier chip made by National Semiconductor; power rating is 4 watts/

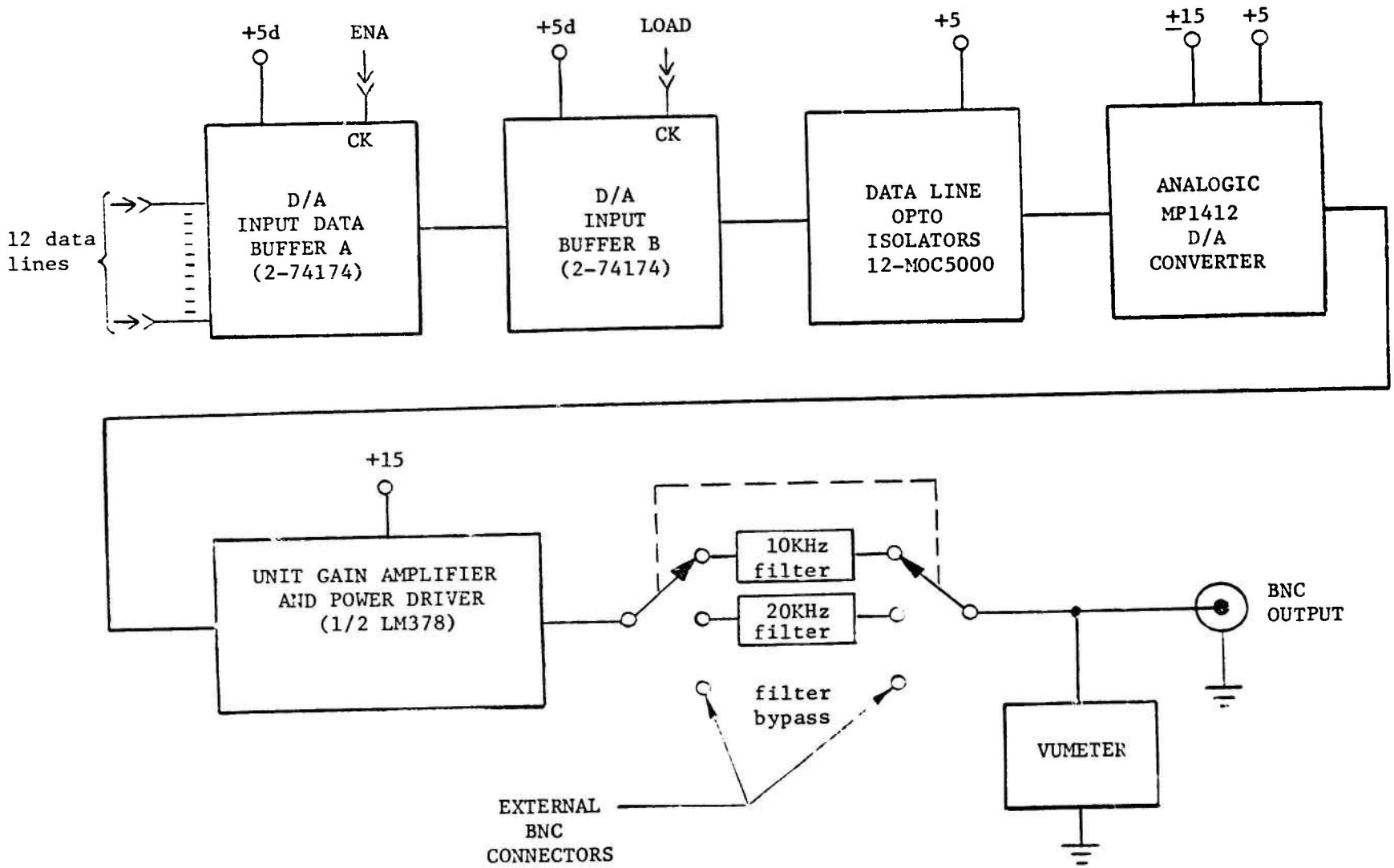


Figure 2-8. Block Diagram of Analog Output Module

channel into 8Ω. Maximum harmonic distortion rating (see Appendix 1) is 1%. This device has great cost and ease of implementation advantages over other approaches. However, it has one bad characteristic, a high frequency oscillation when driving an inductive load (e.g., a speaker coil). Although the oscillation is at too high a frequency to be audible, it is annoying aesthetically from a design point of view and also causes unnecessary power dissipation. Alternative devices are presently being investigated. The expected modification will also allow for DC level output which is not now possible.

The two-stage input buffer provides for synchronous D/A output with asynchronous computer servicing. The converter is always outputting a voltage level corresponding to the contents of the B buffer. The B buffer is reloaded at the 10μsec load command shown in the timing diagram (Figure 2-5). At this point the A buffer contents are moved into the B buffer, so the change in D/A output occurs at intervals equal to the sample time. However, the A buffer contents can be changed by the computer any time in the interval from 10μsec to t_s. During this time the service interrupt is processed and the computer reads the A/D channels and updates the A buffers in the D/A channels. So the double buffer frees the computer from having to service the D/A modules synchronously. Removal of the B buffer would cause a signal distortion due to timing aperture variation. [3]

Figure 2-9 is a photograph of the D/A module output for a sinusoidal input signal. With the system in the TEST mode (generating timing synchronization internally rather than by computer control) the signal, which is shown as the top trace is fed into an A/D module. The digital output of the A/D is fed into a D/A module which reconverts the signal to analog. The bottom trace in Figure 2-9 is the D/A output with the output filters bypassed. The time base is .2ms/cm. The steps in the signal have a length of 1 sample time, which is 40μsec in this case. Notice that this photograph is a representation of the net effect of the two modules on an input signal. It is clear that distortion and noise are not evident on this scale.

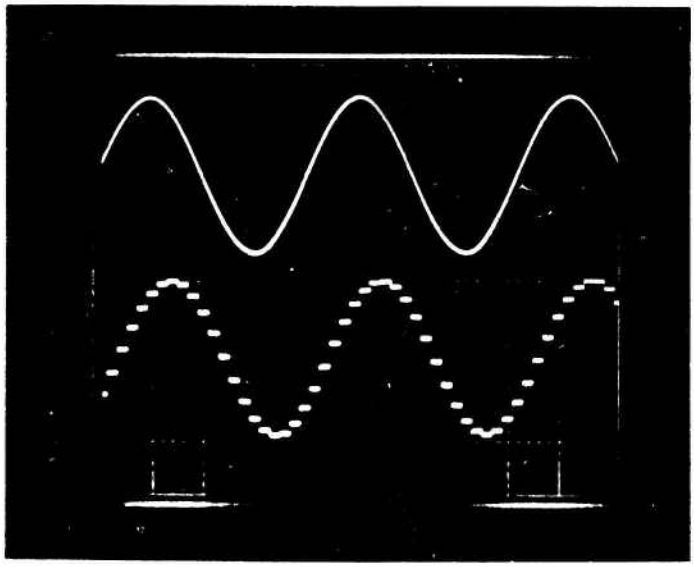


Figure 2-9. Photograph of D/A

2.4 SUMMARY

Experience has shown that care taken in analog signal processing results in greatly increased quality in a digital speech compression system. The increase in quality is a result of decreasing analog broad band device noise, line synchronous noise and ground loop noise. The net effect is an increase in system signal/noise ratio and an increase in dynamic range, the latter resulting because fewer bits are lost to noise. In addition, harmonic and intermodulation distortion can be minimized. It is felt that careful analog processing is a significant factor in the observed quality of the CHI-LPC speech synthesis system.

The two-channel system described in Section 2.2 has been used for all experiments prior to this time, including the generation of the tape enclosed in this report. The multi-channel system has been checked out on the bench and only awaits some software development before implementation. The validity of the design has to a large degree already been proven by the performance of the two-channel system.

Ongoing laboratory study of difference wave equation theory applied to acoustic cavities is based on data acquired through the analog systems described in this chapter. This work provides a good test of the system since signals over the entire audio range can be generated, but with much less structure than typical speech data. Thus, system induced artifacts are more immediately obvious. Results of this work on acoustic cavities will be presented in the next report.

REFERENCES

1. A User's Handbook of Integrated Circuits, E. R. Hnatek, John Wiley and Sons, New York, 1973.

2. Linear Applications Manual, National Semiconductor Corp., 1973.

3. C. M. Tsai, M.S. Thesis, University of Utah, 1973.

2.5 APPENDIX 1

SPECIFICATION PARAMETERS FOR DIGITAL RECORDING SYSTEMS

The purpose of this note is to discuss the definition of quality in digital recording of audio signals. It is suggested here that the utilization of specification parameters familiar to audio engineers is a reasonable first-pass at defining an objective figure of merit. At the same time it must be acknowledged that there are distortion mechanisms in digital/analog systems which do not appear in pure analog systems. The attempt, then, is to compile a useful list of quality parameters which are applicable to digital systems, and then point out those areas in which there are no clearly parameterized definitions of quality.

2.5.1 CANONICAL ANALOG QUALITY PARAMETERS

1. Frequency Response

It is said that the "perfect" audio amplifier is flat within 1.0db from 20Hz to 30Hz, 0.25db from 30Hz to 10KHz, 0.7db from 10KHz to 15KHz and 1.0db from 15KHz to 20KHz.[1] This may be subject to debate, but it illustrates the point that a plot of system frequency response is a valuable specification. For analog systems the store ends at the -3db frequencies, however, for digital systems the character of the high frequency rolloff (db/octave), amount of high frequency leakage and sampling rate must be included to complete the picture.

2. Phase Response

Contrary to the view of Helmholtz and other early researchers, the ear is apparently sensitive to phase differences between frequency bands.[2] As a result it has been suggested that a 17° tolerance in phase shift should be maintained in audio systems.[3]

3. Dynamic Range

Referred to the input of a system, this parameter is the negative db ratio of the input noise voltage to the input signal which causes full scale output. It is useful to be aware of wherein the limitation lies; a clipping amplifier is much more tolerable than an overflowing ADC! (This is also a starting point for discussion of AGC systems.)

4. Input Sensitivity

This parameter is input signal per output voltage. For a microphone the units are sound pressure/output voltage.

5. Hum and Noise

The tolerance of hum and broad band noise is determined to some extent by the listening environment. For example, if $70db_{SPL}$ of signal is being delivered to a room, and if the room ambience is $20db_{SPL}$ (equivalent of a sound studio, not a computer room!), then for the hum and broad band noise to be inaudible, it should be more than 50db below the signal. For high quality audio, the hum and noise should be down about -80db. In the case of digital recording, a reasonable definition of this specification compares quantization noise to analog device noise and hum. If these two are within a factor of $\sqrt{2}$ the system has to be close to optimal. It can also be noted that a small high frequency "glitching" noise component is much more degrading than an audible white component. So in some sense the noise spectrum has to be a consideration here.

6. Distortion

a) Harmonic Distortion

Consider a signal E cos wt. If this is fed through a non-linear system whereby harmonics are generated, then the output signal may have the form (at unit gain),

$$e = E_1 \cos wt + E_2 \cos 2wt + E_3 \cos 3wt + \cdots$$

Here, all components other than E_1 represent harmonic distortion. The accepted parameter for specifying harmonic distortion is %HD;

$$\%HD = \frac{\left(E_2^2+E_3^2+\cdots\right)^{1/2}}{E_1} \times 100.$$

Conventional meters, however, measure

$$\frac{\left(E_2^2+E_3^2+\cdots\right)^{1/2}}{\left(E_1^2+E_2^2+E_3^2+\cdots\right)} \times 100$$

and thus tend to underestimate for large distortion values. Since timbre is just overtone content, some harmonic distortion is tolerable in speech and music; it is effectively masked by the signal. Olson has shown that .75% second harmonic distortion is perceptible in 15KHz music.[4] The higher harmonics, however, are more easily perceived. A threshold of 0.1% and a weighting factor of $n^2/4$ (n=order of harmonic) has been suggested.[5]

b) Intermodulation Distortion

Fortunately for many engineers, a device with some non-linearity will generate sidebands at sums and differences of frequencies appearing at its input. In the linear world this is referred to as inter-modulation distortion. IM can be a more irritating artifact that HD, since the byproducts are not "musically" related to the signal. The threshold for IM perception is said to be .5%.[1] There are two accepted methods for measuring IM distortion:

i. SMPTE method --

Apply the frequencies 60Hz and 7KHz to the input with amplitude ratio 12db. That is, the 60Hz signal is four times larger than the 7KHz signal. Consider the amplitude of the sidebands:

E = 7KHz signal

A_A = 7K+60Hz sideband

A_B = 7K-60Hz sideband

A_C = 7K+120Hz sideband

A_D = 7K-120Hz sideband

Then,

$$\%IMD = \frac{(A_A+A_B)^2+(A_C+A_D)^2}{E_1^2}^{1/2} \times 100.$$

ii. CCIF Method --

Apply two high-frequency signals of equal amplitude, A, and measure the difference frequency amplitude, A_d.

$$\%IMD = \frac{A_d}{2A} \times 100.$$

By sliding both frequencies, maintaining the same difference, the entire frequency band can be measured at the same difference frequency.

c) Transient Intermodulation Distortion [6],[7]

This recently uncovered form of distortion arises from narrow open loop response and heavy negative feedback audio amplifiers. At present, no index exists for its evaluation, however, a qualitative observation can easily be made (see Ref. 7).

2.5.2 DIGITAL ARTIFACTS

Stockham has summarized the effects of digital recording on audio fidelity.[8] Blesser has described an A/D distortion mechanism not mentioned by Stockham; the heterodyne of the input signal with the sample clock.[9] This is apparently also a problem in D/A conversion.[10] At present, there is no standard set of parameters for specifying digital distortion mechanisms. A system for measuring noise and harmonic distortion of converter systems has been developed by Tsai. [11] He suggests use of the log spectrum in characterizing these effects.

REFERENCES

1. Bose Corp., Document #103363-773

2. J. H. Patterson and D. M. Green, J. Acoust. Soc. Am. 48, 894(1970).

3. D. S. Stodolsky, IEEE Trans. Aud, and Elec. Acoust. AU-18, 1970.

4. H. F. Olson, Acoustical Engineering, Van Nostrand, 1957.

5. D. Shorter, Electronic Engineering 22, April 1950.

6. M. Otala, J. Audio Engr. Soc., 20, 396 (1972).

7. W. M. Leach, Audio, Feb. 1975.

8. T. G. Stockham, Jr., Digital Signal Processing. New York, IEEE Press, 1972.

9. B. Blesser, J. Audio Eng. Soc. 22, 20 (1974).

10. J. S. Kritz, IEEE Trans-Acoustics ASSP-23, 146 (1975).

11. C. M. Tsai, M. S. Thesis, University of Utah, 1973.

2.6 APPENDIX 2

PRELIMINARY DATA SHEET FOR MULTI CHANNEL AUDIO SIGNAL SYSTEM

Module	Parameter		
A/D	ANALOG INPUT:	input voltage range	-5V to +5V
		input impedance	1K
		voltage gain	0db to 80db, adjustable
		amplifier bandwidth*	±3db:10Hz to 20KHz
		filter bandwidth	10KHz, 20KHz, bypass
	NOISE:	amplifier equivalent input noise	3μV, 10-20KHz
		quantization noise	.7μV
	DIGITAL OUTPUT:		Tri-State, sinks up to 32mA
	A/D CONVERSION:	conversion time	<10μsec total
		range	-5V to +5V
		resolution	12 bits
		A/D converter	Analogic MP2112
		sample/hold	Analogic MP270
	POWER SUPPLY:	± 15V	±100mA
		+5V	±500mA
		+5V	+500mA
D/A	ANALOG OUTPUT:	output voltage range	-5V to +5V into an 8Ω load
		voltage gain	0db
		bandwidth	±3db:10Hz to 20KHz
		filter bandwidth	10KHz, 20KHz, bypass
	DIGITAL INPUT:		TTL compatible, 1 unit load/line
	D/A CONVERSION:	settling time	5μsec
		range	-5V to +5V
		resolution	12 bits
		D/A converter	Analogic MP1412
	POWER SUPPLY:	±15V	200mA
		±5V	200mA
		+5V	200mA
CONTROL	CLOCK:	period	250nsec or 1 sec
	A/D CONTROL:	sample rate	1 to 4096 clock pulses, variable
		fan out	up to 8 A/D channels
	D/A CONTROL:	fan out	up to 8 D/A channels

*The system will accept analog input down to DC if the amplifier is bypassed via the filter bypass input.